

(19) 日本国特許庁 (J P)

(12) 公開特許公報 (A)

(11) 特許出願公開番号
特開2000-322288
(P2000-322288A)

(43) 公開日 平成12年11月24日 (2000. 11. 24)

(51) Int.Cl. ⁷	識別記号	F I	テマコード [*] (参考)
G 0 6 F 11/28	3 4 0	G 0 6 F 11/28	3 4 0 A 5 B 0 4 2
9/06	5 3 0	9/06	5 3 0 T 5 B 0 4 5
	5 4 0		5 4 0 U 5 B 0 7 6
9/44	5 1 6	9/44	5 1 6
15/16	6 2 0	15/16	6 2 0 T
審査請求 未請求 請求項の数10 O L (全 34 頁)			

(21) 出願番号 特願平11-126471

(22) 出願日 平成11年5月6日 (1999. 5. 6)

(71) 出願人 000005223

富士通株式会社

神奈川県川崎市中原区上小田中4丁目1番
1号

(72) 発明者 久保田 康雄

東京都大田区西蒲田七丁目37番10号 株式
会社富士通金融システムズ内

(72) 発明者 柴 信夫

神奈川県川崎市中原区上小田中4丁目1番
1号 富士通株式会社内

(74) 代理人 100089118

弁理士 酒井 宏明

最終頁に続く

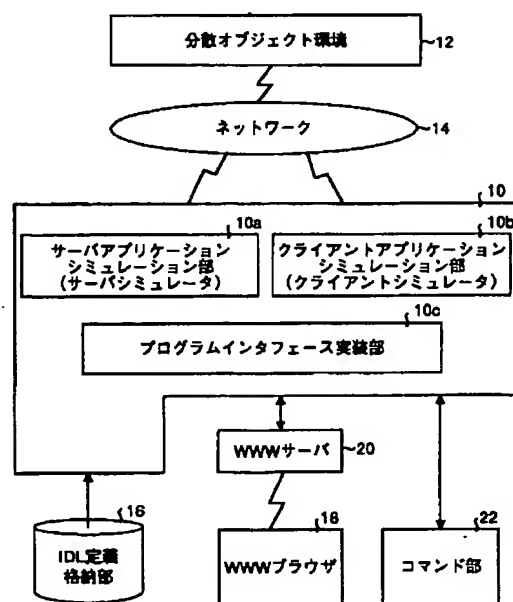
(54) 【発明の名称】 分散オブジェクト開発システム、および、分散オブジェクト開発をコンピュータに実行させるためのプログラムを記録したコンピュータ読み取り可能な記録媒体

(57) 【要約】

【課題】 テスト用アプリケーションの開発を不要にし、かつクライアントおよびサーバアプリケーションの開発環境を分離することができる分散オブジェクト開発システムを提供すること。

【解決手段】 シミュレータマシン10は、IDL定義格納部16からIDL情報を取得し、取得したIDL情報に基づいてサーバシミュレータまたはクライアントシミュレータの機能を自動構築する。また、シミュレータマシン10は、ノードの一つであるWWWサーバ20と接続され、他のノードは、WWWブラウザ18を起動してWWWサーバ20にアクセスすることによって、シミュレータマシン10が提供するサーバシミュレータまたはクライアントシミュレータの機能を自己のノード上に構築することができる。

本発明にかかる分散オブジェクト開発システムの概略構成図



【特許請求の範囲】

【請求項1】 ネットワーク上の複数のノードにオブジェクトが分散された分散オブジェクト環境の下で、前記オブジェクトの呼び出しをおこなう前記ノードの一つであるクライアントにおいて実行されるクライアントアプリケーションと、前記オブジェクトによるサービスを提供する前記ノードの一つであるサーバにおいて実行されるサーバアプリケーションと、を開発する分散オブジェクト開発システムにおいて、

前記オブジェクトの型情報を定義するとともに前記クライアントアプリケーションと前記サーバアプリケーションとの通信を仲介するインタフェース定義情報を取得し、取得した情報に基づいて、前記オブジェクトの呼び出しまたは前記オブジェクトの実行をシミュレートするアプリケーションシミュレート手段を備えたことを特徴とする分散オブジェクト開発システム。

【請求項2】 ネットワーク上の複数のノードにオブジェクトが分散された分散オブジェクト環境の下で、前記オブジェクトの呼び出しをおこなう前記ノードの一つであるクライアントにおいて実行されるクライアントアプリケーションと、前記オブジェクトによるサービスを提供する前記ノードの一つであるサーバにおいて実行されるサーバアプリケーションと、を開発する分散オブジェクト開発システムにおいて、

前記オブジェクトの型情報を定義するとともに前記クライアントアプリケーションと前記サーバアプリケーションとの通信を仲介するインタフェース定義情報を取得し、取得した情報に基づいて、前記クライアントアプリケーションによる前記オブジェクトの呼び出し要求に応じた応答を生成し、生成された応答を前記クライアントアプリケーションに対して送信するサーバアプリケーションシミュレート手段と、

前記インタフェース定義情報を取得し、取得した情報に基づいて、前記オブジェクトの呼び出し要求を生成し、生成された要求を前記サーバアプリケーションに対して送信して、該要求に応じて前記サーバアプリケーションから送信された応答を受信するクライアントアプリケーションシミュレート手段のうち、少なくとも一つを備えたことを特徴とする分散オブジェクト開発システム。

【請求項3】 前記サーバアプリケーションシミュレート手段または前記クライアントアプリケーションシミュレート手段は、前記複数のノードのうちの少なくとも一つのノードに備えられ、当該ノードを除く他のノードのうちの所望のノードが、前記ネットワークを介して、前記サーバアプリケーションシミュレート手段および/または前記クライアントアプリケーションシミュレート手段のシミュレート機能を制御することを特徴とする請求項2に記載の分散オブジェクト開発システム。

【請求項4】 前記サーバアプリケーションシミュレート手段および前記クライアントアプリケーションシミュ

レート手段のシミュレート機能を、前記ノードの一つであるWWW (World Wide Web) サーバ上に備え、前記ノードの他の一つがWWWブラウザを介して前記シミュレート機能を制御することを特徴とする請求項3に記載の分散オブジェクト開発システム。

【請求項5】 前記サーバアプリケーションシミュレート手段は、前記クライアントアプリケーションによる前記オブジェクトの呼び出し要求の内容と、前記生成された応答の内容と、を含む検証用ファイルを作成することを特徴とする請求項2～4のいずれか一つに記載の分散オブジェクト開発システム。

【請求項6】 前記サーバアプリケーションシミュレート手段は、前記クライアントアプリケーションによる前記オブジェクトの呼び出し要求の内容が前記インタフェースの定義に従った型と一致しない場合に、型の不一致を示す情報を前記生成された応答の内容に付加して、前記検証用ファイルを作成することを特徴とする請求項5に記載の分散オブジェクト開発システム。

【請求項7】 前記クライアントアプリケーションシミュレート手段は、前記生成された要求の内容と、該要求に応じて前記サーバアプリケーションから送信された応答の内容と、を含む検証用ファイルを作成することを特徴とする請求項2～4のいずれか一つに記載の分散オブジェクト開発システム。

【請求項8】 前記クライアントアプリケーションシミュレート手段は、前記サーバアプリケーションから送信された応答の内容が前記インタフェースの定義に従った型と一致しない場合に、型の不一致を示す情報を前記生成された要求の内容に付加して、前記検証用ファイルを作成することを特徴とする請求項7に記載の分散オブジェクト開発システム。

【請求項9】 ネットワーク上の複数のノードにオブジェクトが分散された分散オブジェクト環境の下で、前記オブジェクトの呼び出しをおこなう前記ノードの一つであるクライアントにおいて実行されるクライアントアプリケーションと、前記オブジェクトによるサービスを提供する前記ノードの一つであるサーバにおいて実行されるサーバアプリケーションと、を開発するプログラムを記録したコンピュータ読み取り可能な記録媒体であつて、

前記オブジェクトの型情報を定義するとともに前記クライアントアプリケーションと前記サーバアプリケーションとの通信を仲介するインタフェース定義情報を取得するインタフェース情報取得手順と、

取得したインタフェース情報に基づいて、前記オブジェクトの呼び出しおよび/または前記オブジェクトの実行をシミュレートするシミュレート手順と、
を実行させるためのプログラムを記録したコンピュータ読み取り可能な記録媒体。

【請求項10】 ネットワーク上の複数のノードにオブ

ジェクトが分散された分散オブジェクト環境の下で、前記オブジェクトの呼び出しをおこなう前記ノードの一つであるクライアントにおいて実行されるクライアントアプリケーションと、前記オブジェクトによるサービスを提供する前記ノードの一つであるサーバにおいて実行されるサーバアプリケーションと、を開発するプログラムを記録したコンピュータ読み取り可能な記録媒体であって、

前記サーバアプリケーションと前記クライアントアプリケーションのいずれかのシミュレートをおこなうかを選択する選択手順と、

前記オブジェクトの型情報を定義するとともに前記クライアントアプリケーションと前記サーバアプリケーションとの通信を仲介するインタフェース定義情報を取得するインタフェース情報取得手順と、

前記選択ステップにおいて前記サーバアプリケーションのシミュレートをおこなうことが選択された場合、前記インタフェース情報取得手順において取得したインタフェース情報に基づいて、前記クライアントアプリケーションによる前記オブジェクトの呼び出し要求に応じた応答を生成する応答生成手順と、

前記応答生成手順において生成された応答を前記クライアントアプリケーションに対して送信するサーバシミュレート手順と、

前記選択手順において前記クライアントアプリケーションのシミュレートをおこなうことが選択された場合、前記インタフェース情報取得手順において取得したインタフェース情報に基づいて、前記オブジェクトの呼び出し要求を生成する要求生成手順と、

前記要求生成手順において生成された要求を前記サーバアプリケーションに対して送信して、該要求に応じて前記サーバアプリケーションから送信された応答を受信するクライアントシミュレート手順と、

を実行させるためのプログラムを記録したコンピュータ読み取り可能な記録媒体。

【発明の詳細な説明】

【0001】

【発明の属する技術分野】本発明は、ネットワーク上に散りばめられたオブジェクトが相互に連携しながら全体として機能する分散オブジェクトシステムにおいて、特に、オブジェクト指向プログラミングによるクライアント/サーバアプリケーションの開発を支援するための分散オブジェクト開発システム、および、分散オブジェクト開発をコンピュータに実行させるプログラムを記録したコンピュータ読み取り可能な記録媒体に関する。

【0002】

【従来の技術】近年、通信インフラの整備の向上と、LAN (Local Area Network) やインターネットの普及によって、ネットワークを介して所望のサービスを享受することのできる分散コンピューティングの発展が目覚ま

しい。従来からの集中処理型のホストコンピュータ方式においては、データベースの利用や複雑な計算を必要とする処理を高性能なホストコンピュータがおこなうことによって、これに接続された複数の端末が所望のサービスを享受できるという形態であった。

【0003】このホストコンピュータ方式に代わる分散コンピューティングは、近年の低価格かつ高性能なパーソナルコンピュータ (PC) やワークステーション (WS) を用いて、サービスの提供を高信頼性および大量かつ高速処理に向けたサーバ・マシンでおこない、データの加工やユーザインタフェースを上記したPCやWSなどのクライアント・マシンでおこない、さらにこれらをネットワークで接続することにより、処理の分散化を図った、いわゆるクライアント/サーバ・システムを実現したものである。

【0004】分散コンピューティングを実現するネットワークプログラミングは、通常、RPC (リモートプロシジャコール) に基づいておこなわれている。RPCは、ネットワーク上の異なる場所にあるプロシジャ (関数など) の呼び出しをおこなうもので、ネットワークを利用するプロシジャを各サーバにライブラリとして用意しておき、これを通してネットワークの利用を可能とするプログラミングの形態である。

【0005】よって、そのプロシジャのライブラリを整備しておけば、ネットワーク構成、プロトコルおよびプロセス間通信を意識しないでネットワーク・プログラミングをおこなうことが可能となる。このRPCによって、サーバ上のプロシジャがあたかもクライアントにあるかのように見せかけて、ネットワーク間のデータ転送とマシン間のデータ変換を隠蔽することができる。

【0006】ところが、このような分散コンピューティングにおいて、ネットワーク規模の拡大や上記したサービスを提供するためのアプリケーションの高機能化に伴って、通信トラフィックやサーバの負荷の増大、またはアプリケーション開発の複雑化が問題となっている。特にこれまでは、アプリケーションの開発を、オペレーティングシステム (OS) 毎に異なる開発言語や開発ツールなどを使用することを中心に発展させてきたが、ネットワーク化が普及してくると、これまでのアプリケーション開発と、通信、リソース管理、障害対策などの実装環境の構築とを別々におこなってゆくやり方では、アプリケーション開発者に大きな負担と多くの困難を与えることになる。

【0007】そこで、これらの問題を解決するべく、処理単位のさらなる分散化とネットワークプログラミングにおけるアプリケーションコンポーネントの再利用化を図った分散オブジェクト技術が注目されている。ここで、オブジェクトとは、データと処理手続き (メソッドまたはオペレーションと呼ぶ) が一体となったプログラムを意味する。分散オブジェクトシステムでは、ある処

理に対して、オブジェクト同士が動的にクライアント／サーバの関係を構築して処理をおこなうことができ、ネットワーク全体が一つの巨大なコンピュータとして機能することができる。

【0008】上記した分散オブジェクトのアーキテクチャとして代表的なものの一つに、米国のオブジェクト指向技術の標準化団体であるOMG (The Object Management Group) が策定したCORBA (共通オブジェクト・リクエスト・ブローカ・アーキテクチャ) がある。以下に、このCORBAが提供する分散オブジェクト管理の標準仕様について説明する。

【0009】元来、ネットワーク上に分散したアプリケーションコンポーネントを共有して使えるようするためのフレームワークとして、ORB (オブジェクト・リクエスト・ブローカ) という相互コミュニケーションの方式が提案されていた。ORBでは、ネットワーク上のサービスを実行するプログラムをオブジェクトとみなし、クライアントプログラム (すなわちクライアントオブジェクト) は、このORBを介して、同一マシン上またはネットワーク上にあるサーバオブジェクトのメソッドを、それがどこにあるかを意識することなく呼び出すことができる。

【0010】また、ORBは、オブジェクトがネットワーク上のどのマシンに存在するかを管理しており、サービスの呼び出し時にマシン名を指定する必要はないため、ORBを利用したアプリケーションにおいて、あるサービスを別のマシンに移動するような場合でもプログラムの変更を必要とせずにORB上での設定を変えるだけで済むようになり、アプリケーションの柔軟性を向上させることが可能となる。

【0011】よって、クライアントは、ORBを仲介して、利用したいオブジェクトに対してリクエストを送り、リクエストを受け取ったオブジェクトは処理を実行し、再びORBを介して結果をクライアントに返すことになる。ここで、クライアントとは、オブジェクトを利用する側のことであり、サーバは、その利用されるオブジェクト (一般のオブジェクトと区別するため、以下において、オブジェクト・インプリメンテーションと呼ぶ) を提供する側のことである。オブジェクト・インプリメンテーションが別のサーバ・オブジェクトに対するクライアントになることも可能であるため、ORB上のオブジェクトは、その時々でクライアントになったりサーバになったりする。

【0012】また、ORB上の分散オブジェクトはマシンやシステムだけでなく、プログラミング言語にも依存しない形で記述できなければならないため、IDL (インタフェース定義言語) と呼ばれる言語を導入することでこの問題の解決を図っている。IDLは他の言語と違ってインタフェースの定義のみをおこなう専用の言語であり、オペレーション、例外、および属性のセットを宣

言し、各オペレーションは、名前、パラメータ、結果、および例外を定義したシグニチャによって構成される。

【0013】CORBAは、以上に説明したORBの標準仕様であり、特に、クライアントからORBにリクエストを送る際に、クライアント側プログラム (クライアントアプリケーション) の言語 (CやC++) と、ORBを結びつけてメッセージのやり取りを可能にする、いわゆるバインディングの方法として静的起動と動的起動の2種類が用意されていることを特徴の一つとしている。CORBAのオブジェクトを設計するには、最初にIDLを用いてオブジェクトの公開される機能のみを定義し、それを元にして、それぞれのプログラミング言語の実装がおこなわれる。

【0014】図26は、CORBAにおけるORBのインタフェース構造を示すブロック図である。図26に示すCORBAのインタフェース構造は、ORBコア2600、動的起動インタフェース2650、クライアントスタブ2660、ORBインタフェース2670、オブジェクトアダプタ2680、サーバスケルトン2690、インタフェースリポジトリ2630およびインプリメンテーションリポジトリ2640から構成されている。

【0015】図26において、ORBコア2600は、クライアント2610とサーバ (オブジェクト・インプリメンテーション) 2620との間に位置に依存しない通信をつかさどるもので、一般にオブジェクトバスを定義するものとして解釈されている。このオブジェクトバスは、インタフェースリポジトリ2630、インプリメンテーションリポジトリ2640またはその他CORBAサービスによって拡張することができる。

【0016】動的起動インタフェース2650は、上記した動的起動をおこなう際、実行時に、連携する先のオブジェクト (以下、ターゲットオブジェクトと呼ぶ) の型情報を取り出して、ターゲットオブジェクトの呼び出しをおこなうものであり、この動的起動のインタフェース仕様は各オブジェクト共通である。

【0017】また、クライアントスタブ2660は、ターゲットオブジェクトのインタフェース定義がマッピングされた関数であり、クライアント2610がサーバ2620上においてターゲットオブジェクトを起動するための方法を定義するものである。よって、クライアントスタブ2660は、上記した静的起動をおこなうために、あらかじめクライアントアプリケーションの作成時にターゲットオブジェクトの型ごとに生成されて、静的起動をおこなう際、ターゲットオブジェクトの呼び出しをおこなうものである。

【0018】ORBインタフェース2670は、ORBに対する操作を行うためのインタフェースであり、アプリケーションが必要とするローカルなサービスに対する少数のAPI (Application Programming Interface)

から構成され、たとえば、オブジェクトを参照するためのオブジェクト・リファレンスを文字列に変換したり、その逆の変換をおこなうためのAPIを提供している。

【0019】オブジェクトアダプタ2680は、オブジェクト・インプリメンテーションに代わって、サービスの要求を受け付け、オブジェクト・インプリメンテーションの生成、オブジェクト・インプリメンテーションへの要求伝達、オブジェクト・リファレンスを割当てするための実行時環境を提供するものである。特に、ORBでは、ターゲットオブジェクトへの最初の呼び出しがおこなわれると、プロセスが生成される（オブジェクトの活性化）が、オブジェクトアダプタ2680は主にこのオブジェクトの活性化および非活性化をおこなう。

【0020】サーバスケルトン2690は、サーバ2620によってエクスポートされるそれぞれのサービスへの静的なインタフェースの提供、すなわちクライアントスタブ2660が変換した構造を元の関数呼び出し形式に復元して、ターゲットオブジェクトの実装コードの呼び出しをおこなうもので、クライアントスタブ2660とともにIDLコンパイラを使って生成される。

【0021】なお、ここでは、図示していないが、クライアント側の動的起動インタフェース2650と同等の機能をサーバ側に提供する動的スケルトンインタフェースも、上記したサーバスケルトン2690に含むものとする。この動的スケルトンインタフェースは、クライアントアプリケーションの作成時に、ターゲットオブジェクトの実装型が不明な場合に、リクエストをORBからオブジェクト・インプリメンテーションに渡す方法を提供するAPIである。

【0022】インタフェースリポジトリ2630は、動的起動の際に、実行時に取り出されるIDLで作成されたインタフェース情報をオブジェクトの形で格納したものであり、インプリメンテーションリポジトリ2640は、サーバ2620がサポートするクラス、生成されたオブジェクト、およびそれらの実行時リポジトリを提供するものである。

【0023】つぎに、以上に説明したCORBA仕様に準拠した分散オブジェクトシステムにおけるクライアントアプリケーションの開発について説明する。図27は、従来のクライアントアプリケーション開発手順を示すフローチャートである。図27に示すフローチャートは、特に、上記した静的起動をおこなうクライアントアプリケーションの開発をおこなう手順について示している。

【0024】図27において、まず、クライアントアプリケーション開発者は、開発しようとするアプリケーションのプログラム・インタフェース設計をおこなう（ステップS2701）。つづいてクライアントアプリケーションにおいて呼び出される関数のためのIDLファイル、すなわちオブジェクトクラスを作成する（ステップ

S2702）。IDLは、オブジェクトがクライアントの候補に対して、どのようなオペレーションが利用でき、どのようにそれを起動すべきかを教える手段であり、オブジェクトの型、それらの属性、それらがエクスポートするメソッド、およびメソッドのパラメータを定義する。

【0025】そして、上記したオブジェクトクラスを用いて実行される所望のクライアントアプリケーションのソースを作成する（ステップS2703）。ここで、このクライアントアプリケーションが実行される際には、上記したオブジェクトクラスを提供するサーバが必要となるため、ステップS2703において作成したクライアントアプリケーションの実行を検証する場合、すなわちテストする場合には、サーバアプリケーションが必要となる。

【0026】しかしながら、プログラムの実行検証をおこなう段階において、実際にネットワーク上で稼働しているサーバアプリケーションを用いることは、テストされるクライアントアプリケーションによってそのサーバアプリケーションが破壊されたり、ネットワーク上の他のオブジェクトに不具合を与えたりすることが考えられるため、現実的ではない。

【0027】そこで、通常は、テスト用のサーバアプリケーションを作成し（ステップS2704）、このテスト用サーバアプリケーションを起動させた環境において、作成したクライアントアプリケーションの実行検証をおこなう。

【0028】つぎに、ステップS2702において作成したIDLファイルを、IDLコンパイラを用いてコンパイルし、クライアントスタブ、サーバスケルトンおよびIDLオブジェクトを生成する（ステップS2705）。ここで、IDLオブジェクトは、インタフェースリポジトリに結合され（ステップS2706）、動的起動をおこなう際のIDL情報となる。

【0029】そして、ステップS2703において作成されたクライアントアプリケーションをコンパイルしてオブジェクトを生成し、このオブジェクトに、ステップS2705において生成されたクライアントスタブをリンクすることで、実行可能なクライアントアプリケーションのファイルを得る（ステップS2707）。

【0030】また、同様に、ステップS2704において作成されたテスト用サーバアプリケーションをコンパイルしてオブジェクトを生成し、このオブジェクトに、ステップS2705において生成されたサーバスケルトンをリンクすることにより、実行可能なテスト用サーバアプリケーションのファイルを得る（ステップS2708）。

【0031】ステップS2707によって得たテスト用サーバアプリケーションは、インプリメンテーションリポジトリに記録されるとともに、クライアントアプリケ

ーションによって呼び出し可能なオブジェクト・インプリメンテーションとして登録される(ステップS2709)。

【0032】そして、作成したクライアントアプリケーションの実行検証をおこなうために、クライアントから送信されたリクエストに対して応答する電文(テキストイメージ)となるテストデータを作成し(ステップS2710)、ステップS2709において登録されたテスト用サーバアプリケーションの起動をおこなう(ステップS2711)。この状態で、ステップS2707において生成されたクライアントアプリケーションを実行し、その結果を検証する(ステップS2712)。

【0033】一方、サーバアプリケーションの開発をおこなう場合も上述したクライアントアプリケーションの開発と同様に、テスト用クライアントアプリケーションを作成する必要がある。図28は、従来のサーバアプリケーション開発手順を示すフローチャートである。図28に示すフローチャートにおいても、静的起動をおこなうサーバアプリケーションの開発をおこなう手順について示している。

【0034】図28において、まず、サーバアプリケーション開発者は、開発しようとするサーバアプリケーションのプログラム・インタフェース設計をおこなう(ステップS2801)。つづいてサーバアプリケーションにおいて呼び出される関数のためのIDLファイル、すなわちオブジェクトクラスを作成する(ステップS2802)。

【0035】そして、上記したオブジェクトクラスを用いて実行される所望のサーバアプリケーションのソースと、テスト用のクライアントアプリケーションのソースとを作成する(ステップS2803、ステップS2804)。つづいて、ステップS2802において作成したIDLファイルを、IDLコンパイラを用いてコンパイルし、クライアントスタブ、サーバスケルトンおよびIDLオブジェクトを生成し(ステップS2805)、IDLオブジェクトを、インタフェースリポジトリに結合する(ステップS2806)。

【0036】そして、ステップS2803において作成されたサーバアプリケーションをコンパイルしてオブジェクトを生成し、このオブジェクトに、ステップS2805において生成されたサーバスケルトンをリンクすることにより、実行可能なサーバアプリケーションのファイルを得る(ステップS2807)。

【0037】また、同様に、ステップS2804において作成されたテスト用クライアントアプリケーションをコンパイルしてオブジェクトを生成し、このオブジェクトに、ステップS2805において生成されたクライアントスタブをリンクすることで、実行可能なテスト用クライアントアプリケーションのファイルを得る(ステップS2808)。

【0038】ステップS2808において得られたサーバアプリケーションは、インプリメンテーションリポジトリに記録されるとともに、テスト用クライアントアプリケーションによって呼び出し可能なオブジェクト・インプリメンテーションとして登録される(ステップS2809)。

【0039】そして、サーバアプリケーションの実行検証をおこなうために、テスト用クライアントアプリケーションからサーバアプリケーションに送信されるリクエストとなるテストデータ(テキストイメージ)を作成し(ステップS2810)、ステップS2809において登録されたサーバアプリケーションを起動する(ステップS2811)。この状態で、ステップS2807において生成されたテスト用クライアントアプリケーションを実行し、その実行に応じて得られるサーバアプリケーションの動作結果を検証する(ステップS2812)。

【0040】以上に説明したように、分散オブジェクトシステムにおいては、クライアント/サーバ・モデルを基本構造としているため、通常、アプリケーションの実行において、対となるクライアントまたはサーバを必要とし、アプリケーションの開発をおこなう段階にあっても、作成したアプリケーションの実行検証をするために、対となるクライアントまたはサーバ上で動作するアプリケーションが必要となる。

【0041】また、他の分散オブジェクト・システムにおけるアプリケーションの開発方法として、たとえば特開平9-120366号公報に開示の「分散アプリケーション・プログラムをデバッグする分散デバッグのためのシステムおよび方法」が提案されている。この「分散アプリケーション・プログラムをデバッグする分散デバッグのためのシステムおよび方法」によれば、CORBA準拠の分散オブジェクト環境において、ローカル・コンピュータおよび遠隔コンピュータにデバッグ・エンジンを常駐させ、これらローカル・コンピュータおよび遠隔コンピュータのうちの少なくとも一つにデバッグGUIを備えている。

【0042】そして、このデバッグGUIが通信機構によってデバッグ・エンジンと通信することにより、クライアントアプリケーションがローカル・ホストでデバッグを使用できるとともに、オブジェクトを含んでいるアプリケーションおよび未知の遠隔ホスト・コンピュータで動作するオブジェクト・インプリメンテーションをシームレスにデバッグすることが可能となっている。

【0043】

【発明が解決しようとする課題】しかしながら、CORBAに準拠した分散オブジェクトシステムでは、通信相手となるアプリケーション(オブジェクト)の場所およびインタフェースを認識しているのは、開発者ではなくORBであり、クライアントアプリケーション開発者がサーバアプリケーション開発者でもあるという特別な場

合を除いては、未知のホストシステム上に配置される未知のオブジェクトとの通信が要求される。

【0044】すなわち、CORBA準拠システムにおいては、クライアントアプリケーションがターゲットオブジェクトの配置位置を直接取得することなく、ターゲットオブジェクトの使用を可能にしていることを特徴としており、一般に、CORBA準拠システムのアプリケーション開発者は自分のオブジェクトに関連したサーバを探し出すことができない。

【0045】よって、CORBA仕様に準拠するような分散オブジェクトシステム上で動作させるアプリケーションを開発するには、上述したように、目的とするクライアントアプリケーションまたはサーバアプリケーションのソース作成とは別に、テスト用のクライアントまたはサーバアプリケーションを作成する必要があった。このことは、目的とするアプリケーションの実行検証の信頼性を高めるためにも、テスト用のアプリケーションにバグが含まれることは許されないため、アプリケーション開発者に、多大な負担を与える結果となっていた。

【0046】また、CORBA仕様に準拠する分散オブジェクトシステムは、アプリケーションの言語依存性、OS依存性を排除するために上述したIDLを導入している一方で、アプリケーション開発者にアプリケーションのプログラム言語とは別にIDLという言語の習得を要請するものであり、作成したアプリケーションの実行検証においてもこのIDLの入出力が正しくおこなわれているかどうかを検証する必要があるが、図27および図28に示したような従来のアプリケーション開発手順では、クライアントおよびサーバの開発環境が分離されてい

ないため、ネットワークを介した実際の分散オブジェクト環境上、特にORB上におけるアプリケーションの動作を完全に再現することはできず、実行検証をおこなうシステム（ホストシステム）に依存したアーキテクチャ（OS等）に従って、クライアントのリクエストまたはサーバの応答を作成しなければならなかった。

【0047】また、上記した特開平9-120366号公報に開示の「分散アプリケーション・プログラムをデバッグする分散デバッグのためのシステムおよび方法」では、デバッグ環境を実際の分散オブジェクト環境上で実現しているために、より信憑性の高いアプリケーションの実行検証をおこなうことができる一方で、バグの存在によって、ネットワークに接続された他のノード（他のクライアントやサーバ）に影響を与える可能性がある。

【0048】さらに、実行検証させるサーバまたはクライアントアプリケーションと対になるサーバまたはクライアントアプリケーションのマシンにおいてもデバッグ・マシンを搭載する必要があるが、CORBA仕様のよう

に、所望のホストシステム上にアプリケーション開発環境を構築することができる分散オブジェクト開発システム、および、分散オブジェクト開発をコンピュータに実行させるプログラムを記録したコンピュータ読み取り可能な記録媒体を提供することを目的とする。

【0049】本発明は、上記に鑑みてなされたものであって、サーバアプリケーションに対してクライアントアプリケーション、クライアントアプリケーションに対してサーバアプリケーションのように、実行検証の対象となるアプリケーションに対して対となるアプリケーションをシミュレートする各シミュレータを備え、テスト用アプリケーションの開発を不要にし、クライアントおよびサーバアプリケーションの開発環境を分離するとともに、所望のホストシステム上にアプリケーション開発環境を構築することができる分散オブジェクト開発システム、および、分散オブジェクト開発をコンピュータに実行させるプログラムを記録したコンピュータ読み取り可能な記録媒体を提供することを目的とする。

【0050】

【課題を解決するための手段】上述した課題を解決し、目的を達成するために、請求項1の発明において、ネットワーク上の複数のノードにオブジェクトが分散された分散オブジェクト環境の下で、前記オブジェクトの呼び出しをおこなう前記ノードの一つであるクライアントにおいて実行されるクライアントアプリケーションと、前記オブジェクトによるサービスを提供する前記ノードの一つであるサーバにおいて実行されるサーバアプリケーションと、を開発する分散オブジェクト開発システムは、前記オブジェクトの型情報を定義するとともに前記クライアントアプリケーションと前記サーバアプリケーションとの通信を仲介するインタフェース定義情報を取得し、取得した情報に基づいて、前記オブジェクトの呼び出しまたは前記オブジェクトの実行をシミュレートするアプリケーションシミュレート手段（後述するサーバシミュレータまたはクライアントシミュレータに相当する）を備えているので、アプリケーションの実行検証をおこなう際に対となるテスト用のアプリケーションの実行に必要なテストデータ（後述する情報ファイルに相当する）の自動生成が可能となり、このテストデータをオブジェクトの呼び出しおよび/またはオブジェクトの実行をおこなう際の通信電文として用いることで、サーバシミュレータまたはクライアントシミュレータを自動的に構築することができる。

【0051】また、請求項2の発明において、ネットワーク上の複数のノードにオブジェクトが分散された分散オブジェクト環境の下で、前記オブジェクトの呼び出しをおこなう前記ノードの一つであるクライアントにおいて実行されるクライアントアプリケーションと、前記オブジェクトによるサービスを提供する前記ノードの一つであるサーバにおいて実行されるサーバアプリケーションと、を開発する分散オブジェクト開発システムは、前記オブジェクトの型情報を定義するとともに前記クライアントアプリケーションと前記サーバアプリケーションとの通信を仲介するインタフェース定義情報を取得し、

取得した情報に基づいて、前記クライアントアプリケーションによる前記オブジェクトの呼び出し要求に応じた応答（後述する復帰情報に相当する）を生成し、生成された応答を前記クライアントアプリケーションに対して送信するサーバアプリケーションシミュレート手段（後述するサーバシミュレータに相当する）を備えているので、サーバアプリケーションをシミュレートする機能を自動的に構築することができる。

【0052】また、前記インタフェース定義情報を取得し、取得した情報に基づいて、前記オブジェクトの呼び出し要求（後述する起動情報に相当する）を生成し、生成された要求を前記サーバアプリケーションに対して送信して、該要求に応じて前記サーバアプリケーションから送信された応答を受信するクライアントアプリケーションシミュレート手段（後述するクライアントシミュレータに相当する）を備えているので、クライアントアプリケーションをシミュレートする機能を自動的に構築することができる。

【0053】また、請求項3の発明において、請求項2に記載の分散オブジェクト開発システムは、前記複数のノードのうちの少なくとも一つのノードが前記サーバアプリケーションシミュレート手段（後述するサーバシミュレータに相当する）または前記クライアントアプリケーションシミュレート手段（後述するクライアントシミュレータに相当する）を備えており、当該ノードを除く他のノードのうちの所望のノードが、前記ネットワークを介して、前記サーバアプリケーションシミュレート手段および／または前記クライアントアプリケーションシミュレート手段のシミュレート機能を制御するので、テストの対象となるアプリケーションと、シミュレートされるアプリケーションとをネットワーク上に分離して配置することができる。

【0054】また、請求項4の発明において、請求項3に記載の分散オブジェクト開発システムは、前記サーバアプリケーションシミュレート手段（後述するサーバシミュレータに相当する）および前記クライアントアプリケーションシミュレート手段（後述するクライアントシミュレータに相当する）のシミュレート機能（後述するシミュレートプログラム）を、前記ノードの一つであるWWW（World Wide Web）サーバ上に備え、前記ノードの他の一つがWWWブラウザを介して前記シミュレート機能を制御するので、WWWブラウザ上のGUI表示による操作を提供することができる。

【0055】また、請求項5の発明において、請求項2～4のいずれか一つに記載の分散オブジェクト開発システムは、前記サーバアプリケーションシミュレート手段（後述するサーバシミュレータに相当する）が、前記クライアントアプリケーションによる前記オブジェクトの呼び出し要求の内容（後述する起動情報に相当する）と、前記生成された応答の内容（後述する実行履歴に相

当する）と、を含む検証用ファイルを作成するので、この検証用ファイルを参照することにより、開発したクライアントアプリケーションの設計の誤りを検証することができる。

【0056】また、請求項6の発明において、請求項5に記載の分散オブジェクト開発システムは、前記サーバアプリケーションシミュレート手段（後述するサーバシミュレータに相当する）が、前記クライアントアプリケーションによる前記オブジェクトの呼び出し要求の内容（後述する起動情報に相当する）が前記インタフェースの定義に従った型と一致しない場合に、型の不一致を示す情報を前記生成された応答の内容（後述する実行履歴に相当する）に付加して、前記検証用ファイルを作成するので、検証用ファイルを参照することで、開発したクライアントアプリケーションの設計の誤りを容易に発見することができる。

【0057】また、請求項7の発明において、請求項2～4のいずれか一つに記載の分散オブジェクト開発システムは、前記クライアントアプリケーションシミュレート手段（後述するクライアントシミュレータに相当する）が、前記生成された要求の内容（後述する実行履歴に相当する）と、該要求に応じて前記サーバアプリケーションから送信された応答の内容（後述する復帰情報に相当する）と、を含む検証用ファイルを作成するので、この検証用ファイルを参照することにより、開発したサーバアプリケーションの設計の誤りを検証することができる。

【0058】また、請求項8の発明において、請求項7に記載の分散オブジェクト開発システムは、前記クライアントアプリケーションシミュレート手段が、前記サーバアプリケーションから返信された応答の内容（後述する復帰情報に相当する）が前記インタフェースの定義に従った型と一致しない場合に、型の不一致を示す情報を前記生成された要求の内容（後述する実行履歴に相当する）に付加して、前記検証用ファイルを作成するので、開発したサーバアプリケーションの設計の誤りを容易に発見することができる。

【0059】また、請求項9の発明において、ネットワーク上の複数のノードにオブジェクトが分散された分散オブジェクト環境の下で、前記オブジェクトの呼び出しをおこなう前記ノードの一つであるクライアントにおいて実行されるクライアントアプリケーションと、前記オブジェクトによるサービスを提供する前記ノードの一つであるサーバにおいて実行されるサーバアプリケーションと、を開発するプログラムを記録したコンピュータ読み取り可能な記録媒体は、前記オブジェクトの型情報を定義するとともに前記クライアントアプリケーションと前記サーバアプリケーションとの通信を仲介するインタフェース定義情報を取得するインタフェース情報取得手順と、取得したインタフェース情報に基づいて、前記オ

ブジェクトの呼び出しおよび／または前記オブジェクトの実行をシミュレートするシミュレート手順と、を実行させるためのプログラムを記録しているので、サーバアプリケーションをシミュレートする機能を自動的に構築することができる。

【0060】また、請求項10の発明において、ネットワーク上の複数のノードにオブジェクトが分散された分散オブジェクト環境の下で、前記オブジェクトの呼び出しをおこなう前記ノードの一つであるクライアントにおいて実行されるクライアントアプリケーションと、前記オブジェクトによるサービスを提供する前記ノードの一つであるサーバにおいて実行されるサーバアプリケーションと、を開発するプログラムを記録したコンピュータ読み取り可能な記録媒体は、前記サーバアプリケーションと前記クライアントアプリケーションのいずれかのシミュレートをおこなうかを選択する選択手順と、前記オブジェクトの型情報を定義するとともに前記クライアントアプリケーションと前記サーバアプリケーションとの通信を仲介するインタフェース定義情報を取得するインタフェース情報取得手順と、を実行させるためのプログラムを記録しているので、サーバアプリケーションをシミュレートする機能を自動的に構築することができる。

【0061】また、前記選択手順において前記サーバアプリケーションのシミュレートをおこなうことが選択された場合、前記インタフェース情報取得手順において取得したインタフェース情報に基づいて、前記クライアントアプリケーションによる前記オブジェクトの呼び出し要求に応じた応答を生成する応答生成手順と、前記応答生成手順において生成された応答を前記クライアントアプリケーションに対して送信するサーバシミュレート手順と、前記選択手順において前記クライアントアプリケーションのシミュレートをおこなうことが選択された場合、前記インタフェース情報取得手順において取得したインタフェース情報に基づいて、前記オブジェクトの呼び出し要求を生成する要求生成手順と、前記要求生成手順において生成された要求を前記サーバアプリケーションに対して送信して、該要求に応じて前記サーバアプリケーションから送信された応答を受信するクライアントシミュレート手順と、を実行させるためのプログラムを記録しているので、サーバアプリケーションをシミュレートする機能を自動的に構築することができる。

【0062】

【発明の実施の形態】以下に、分散オブジェクト開発システム、および、分散オブジェクト開発をコンピュータに実行させるためのプログラムを記録したコンピュータ読み取り可能な記録媒体の実施の形態を図面に基いて詳細に説明する。なお、この実施の形態によりこの発明が限定されるものではない。

【0063】図1は、本発明にかかる分散オブジェクト開発システムの概略構成を示すブロック図である。図1

において、分散オブジェクト開発システムは、OMG仕様によるORB間通信用のネットワークプロトコルであるIIOP等の通信プロトコルによって構築されたネットワーク14に、それぞれオブジェクトを分散して備えた複数のノード（PC、WS、ホストコンピュータ等）が接続されることで構築された、CORBA等に準拠した分散オブジェクト環境12の下にあって、上記したノードの一つとしてネットワーク14に接続されたシミュレータマシン10と、シミュレータマシン10においてクライアントシミュレータおよびサーバシミュレータの機能を実現するために用いられるIDL定義格納部16と、を備えて構成されている。

【0064】シミュレータマシン10は、さらに、IDL定義格納部16から取得されるIDL情報に基づいて自動生成されるテスト用サーバアプリケーション、すなわちサーバシミュレータの機能を提供するサーバアプリケーションシミュレーション部10aと、IDL定義格納部16から取得されるIDL情報に基づいて自動生成されるテスト用クライアントアプリケーション、すなわちクライアントシミュレータの機能を提供するクライアントアプリケーションシミュレーション部10bと、上記したテスト用サーバアプリケーションおよびテスト用クライアントアプリケーションの自動構築およびシミュレータマシン10に接続される外部装置（上記したノードを含む）の通信制御を担うプログラムインタフェース実装部10cと、を備えている。

【0065】ここで、IDL定義とは、上述したIDL情報を意味し、クライアントアプリケーションおよびサーバアプリケーションのインタフェースを定義するものであり、IDL定義格納部16は、上記したノードの一つとしてネットワーク14上に配置されるか、またはシミュレータマシン10の外部デバイスとして接続されたものである。

【0066】また、シミュレータマシン10は、ユーザによってコマンド部22を介して操作され、このコマンド部22によって与えられた種々の命令に従って動作することが可能である。さらに、シミュレータマシン10は、ノードの一つであるWWW（World Wide Web）サーバ20と接続されており、他のノード上で起動されるWWWブラウザ18がこのWWWサーバ20にアクセスすることにより、WWWブラウザ18上に、シミュレータマシン10が提供する機能（サーバシミュレータ、クライアントシミュレータ）を享受することができる。

【0067】すなわち、シミュレータマシン10は、コマンド部22以外にも、遠隔にあるノードからネットワーク14を介して操作することができる。特に、WWWブラウザ18を利用する場合、上記した機能は、WWWサーバ20によってWWWブラウザ18上にサービスを提供する外部プログラムとして実現される。この場合、WWWサーバ20は、CGI（Common Gateway Interfa

、ce)やAPIを経由することによって外部プログラム、すなわちシミュレータマシン10の機能を利用することができる。

【0068】よって、本発明にかかる分散オブジェクト開発システムにおいて、シミュレータマシン10は、アプリケーション開発者がクライアントアプリケーションの開発をおこなう際には、そのクライアントアプリケーションの実行検証をおこなうために必要なテスト用サーバアプリケーションをシミュレートし、サーバアプリケーションの開発をおこなう際には、そのサーバアプリケーションの実行検証をおこなうために必要なテスト用クライアントアプリケーションをシミュレートすることができる。

【0069】さらに、このシミュレート機能は、WWWサーバ20を介して、所望のノードから実行することができる。このことは、インターネットのような広範囲に亘るネットワークに接続可能なノード上において、アプリケーションの開発をおこなうことができ、アプリケーション開発者が使用するノード毎に上記したシミュレート機能を有するプログラムをインストールする必要がな

くなることを意味する。

【0070】また、分散オブジェクト環境において動作するアプリケーションのように、本来、分散されたノードのいずれかに配置されたターゲットオブジェクトを呼び出すアプリケーション、すなわちクライアントアプリケーションや、ターゲットオブジェクトを備えてクライアントアプリケーションから呼び出されるアプリケーション、すなわちサーバアプリケーションについての実行検証をおこなうには、ネットワークを介して遠隔にあるノードとの通信が必要となるが、本発明にかかる分散オブジェクト開発システムでは、IDL定義格納部16からターゲットオブジェクトについてのIDL情報を取得することにより、そのターゲットオブジェクトによるリクエストの送受信をシミュレートしたテスト用アプリケーションを自己のノードに自動的に構築することができる。

【0071】以下において、この分散オブジェクト開発システムを用いて、クライアントアプリケーションおよびサーバアプリケーションの開発手順について説明し、各アプリケーションの開発をおこなうにあたり、上記したサーバアプリケーションシミュレーション部（以下、サーバシミュレータと呼ぶ）およびクライアントアプリケーションシミュレーション部（以下、クライアントシミュレータと呼ぶ）の処理についてそれぞれ、WWWブラウザ18上で実現した場合に表示される画面とともに説明する。

【0072】（クライアントアプリケーションの開発手順）まず、図27に示した従来のクライアントアプリケーション開発手順に対応させて、本発明にかかる分散オブジェクト開発システムにおけるサーバシミュレータを

用いたクライアントアプリケーションの開発手順について説明する。図2は、本発明にかかる分散オブジェクト開発システムによるクライアントアプリケーション開発手順を示すフローチャートである。図2に示すフローチャートは、特に、上記した静的起動をおこなうクライアントアプリケーションの開発をおこなう手順について示している。

【0073】図2において、まず、クライアントアプリケーション開発者は、開発しようとするアプリケーションのプログラム・インタフェース設計をおこなう（ステップS201）。つづいてクライアントアプリケーションにおいて呼び出される関数のためのIDLファイルを作成する（ステップS202）。

【0074】そして、上記したIDLファイルを用いて実行される所望のクライアントアプリケーションのソースを作成する（ステップS203）。つぎに、ステップS202において作成したIDLファイルを、IDLコンパイラを用いてコンパイルし、クライアントスタブ、サーバスケルトンおよびIDLオブジェクトを生成する（ステップS204）。生成されたIDLオブジェクトは、インタフェースリポジトリに結合される（ステップS205）。

【0075】つづいて、ステップS203において作成されたクライアントアプリケーションのソースをコンパイルしてオブジェクトを生成し、このオブジェクトに、ステップS204において生成されたクライアントスタブをリンクすることで、実行可能なクライアントアプリケーションのファイルを得る（ステップS206）。

【0076】そして、上記したサーバシミュレータを起動する前処理を実行し（ステップS207）、その後、サーバシミュレータを起動する処理を実行する（ステップS208）。これらサーバシミュレータ起動前処理およびサーバシミュレータ起動処理が、本発明にかかる分散オブジェクト開発システムによって実現される処理に相当し、これら処理の詳細については後述する。

【0077】サーバシミュレータを起動させた後は、ステップS206において実行可能なファイルとして生成されたクライアントアプリケーションを実行させる（ステップS209）。ここで実行されるクライアントアプリケーションは、分散オブジェクト環境に実際に搭載する形態と同一のコードを有するものであり、特にテストのために変更を加える必要はない。すなわち、CORBA上で問題なく動作可能なバイナリソースのクライアントアプリケーションを実行させる。そして、クライアントアプリケーションの実行結果の検証をおこなう（ステップS210）。

【0078】図3は、このクライアントアプリケーションの実行処理を説明するためのフローチャートであり、特に静的起動をおこなうためのクライアントスタブがおこなう処理を示している。図3において、まずクライ

10

20

30

40

50

ントアプリケーションは、初期化がおこなわれ、必要に応じてオブジェクトの呼び出しリクエストをクライアントスタブに送出する(ステップS301)。CORBA仕様では、このオブジェクト、すなわちターゲットオブジェクトの呼び出しをすべてORB上のオブジェクト・リファレンスと呼ばれる参照IDによっておこなっているため、このターゲットオブジェクトのオブジェクト・リファレンスを取得する必要がある。

【0079】しかし、クライアントアプリケーションは、ターゲットオブジェクトを文字列からなるオブジェクト名を用いて指定しているため、このオブジェクト名に対応したオブジェクト・リファレンスを取得しなければならない。CORBAには、オブジェクト名からそれに対応したオブジェクト・リファレンスを提供するネーミングサービスが備わっているため、通常は、このネーミングサービスを利用する。

【0080】ただし、ネーミングサービスもまたCORBA上のオブジェクトの一つであるため、あらかじめこのネーミングサービスのオブジェクト・リファレンスを取得する必要がある(ステップS302)。これにより、ネーミングサービスへのアクセスが可能となり、ターゲットオブジェクト(広義ではサーバアプリケーション)のオブジェクト・リファレンスの取得が可能となる(ステップS303)。

【0081】そして、クライアントスタブは、ステップS303において取得したターゲットオブジェクトのオブジェクト・リファレンスを用いて、ORBコアに送出するリクエストを生成する(ステップS304)。オブジェクトアダプタは、このリクエストを受け取り、リクエストに含まれるターゲットオブジェクトに対応するサーバスケルトンによって、オブジェクトの呼び出しがおこなわれる(ステップS305)。

【0082】実際には、ステップS305の後、サーバスケルトンが、呼び出したオブジェクトから返信データを受け取り、ORBコアを介して再び、リクエスト発信元のクライアントに対してその返信データを送出するが、本発明にかかる分散オブジェクト開発システムでは、この処理を、上記したサーバシミュレータがおこなう。よって、サーバシミュレータは、このサーバスケルトンの機能をシミュレートすることにもなる。

【0083】図3に示したクライアントアプリケーションが実行された後は、サーバシミュレータにより、クライアントに対して返信データが送出され、クライアントアプリケーションの動作結果の検証がおこなわれる(図2のステップS210)。具体的には、クライアントアプリケーションが送信したリクエストと、サーバシミュレータが受信したデータとを比較することにより、クライアントアプリケーションの設計の誤りを検出する。

【0084】(サーバシミュレータの動作) つぎに、サーバシミュレータの動作について説明する。このサーバ

シミュレータの動作は、上述したサーバシミュレータ起動前処理およびサーバシミュレータ起動処理に相当するものである。図4は、本発明にかかる分散オブジェクト開発システムのサーバシミュレータの動作を示すフローチャートである。特に、ここでは、WWWブラウザ18上でサーバシミュレータの制御をおこなう場合について説明する。

【0085】図4において、まず、アプリケーション開発者は、分散オブジェクト環境12の下にあるノードの一つ(多くはパーソナルコンピュータである)を使用して、WWWブラウザ18を起動させる。そして、シミュレータマシン10の機能を実現するプログラム(以下、シミュレートプログラムと呼ぶ)をHTML(HyperText Markup Language)として提供するWWWサーバ20にアクセスするために、そのWWWサーバ20のアドレスを示すURL(Uniform Resource Locator)をWWWブラウザ18に入力する。これにより、WWWブラウザ18上に、シミュレータマシン10の機能を選択するためのメニュープログラムが起動する(ステップS401)。

【0086】図5は、ステップS401において起動されたメニュープログラムの起動メニュー画面を示す説明図である。図5において、上記したURLは、アドレス入力フィールド505に入力される。図中においては、「http://test/bin/menu.cgi」が入力された状態が示されており、特に、シミュレートプログラムがCGIを介した外部プログラムとしてWWWブラウザ18と接続されることがわかる。

【0087】また、図5に示すWWWブラウザ18では、流通している従来のWWWブラウザのGUI(Graphical User Interface)と同様に、WWWページ戻りボタン501、WWWページ進むボタン502、アクセス中止ボタン503およびWWWページ更新ボタン504等が配置されており、マウス等のポインティングデバイスを用いることで各ボタンやその他の選択項目を選択および確定することができ、また、キーボードを用いることで文字入力を可能とする。

【0088】図5に示すように、シミュレートプログラムによって提供されるメニューは、後述する情報データの作成をおこなうための「Simulation Data Editor」510、クライアントシミュレータを起動するための「Client Simulator」520およびサーバシミュレータを起動するための「Server Simulator」530の選択項目から構成される。

【0089】なお、これら選択項目は、各項目に応じた処理を示すページへとリンクされており、WWWブラウザ18の表示画面において、実際にはこのようなページの更新がおこなわれる毎に、アドレス入力フィールド505に表示されるアドレスが変更されるが、以下の説明において参照される表示画面中では、そのアドレスの表

10

20

30

40

50

示を省略する。

【0090】本発明にかかる分散オブジェクト開発システムにおいては、サーバシミュレータを起動させる前に、そのサーバシミュレータを自動的に構築するために、情報ファイルの作成を必要としているため、まず、図5に示した画面上において、「Simulation Data Editor」510を選択する。

【0091】情報ファイルの作成処理は、サーバシミュレータとクライアントシミュレータで共通の入力フォームを与えるため、ここでは、どちらのシミュレータにおける情報ファイルの作成をおこなうかを指示する必要がある。この指示は、図5において、「Simulation Data Editor」510を2つの選択項目、たとえば、「Client Simulation Data Editor」と「Server Simulation Data Editor」とに分けて表示し、これら2者間の選択をおこなうようにしてもよいし、「Simulation Data Editor」510の選択確定後に、新たにサーバシミュレータモードかクライアントシミュレータモードかの選択を促す選択項目を表示してもよく、設計上適宜変更できるものである。

【0092】ここでは、サーバシミュレータの起動を目的とするため、サーバシミュレータモードの選択をおこなうものとする(ステップS402)。この選択により、復帰情報作成処理へと処理が移行する(ステップS403)。ここで、復帰情報とは、上記した情報ファイルをサーバシミュレータモードにおいて作成する場合のその情報ファイルの呼び名である。具体的には、この復帰情報は、サーバからクライアントへ送信される返信情報を示し、この場合では、サーバシミュレータがクライアントアプリケーションに対して返信する情報を意味する。

【0093】(復帰情報作成処理)ここで、ステップS403における復帰情報作成処理について説明する。この復帰情報作成処理は、図2のステップS207におけるサーバシミュレータ起動前処理に相当する処理でもある。図6は、ステップS403におけるサーバシミュレータの復帰情報作成処理を示すフローチャートである。また、図7は、情報ファイル(ここでは復帰情報を意味する)作成処理における初期入力画面を示す説明図である。

【0094】図7に示す画面上において、まず、情報ファイル名入力フィールド710に、復帰情報ファイル名を入力する(ステップS601)。この復帰情報ファイル名とは、これから作成する復帰情報の格納先となるファイルの名前であり、たとえば「/home/apts/dat/apt0001_sv.dat」のようなものである。また、この復帰情報ファイル名は、後述するサーバシミュレータの起動時において参照される。

【0095】つづいて、オブジェクト名入力フィールド720に、サーバオブジェクト名を入力する(ステップ

S602)。このオブジェクト名とは、クライアントアプリケーションによって呼び出され、かつサーバシミュレータ上でシミュレートされるターゲットオブジェクトの名前であり、たとえば「IDL:apts0001/apts0001:1.0」のようなものである。

【0096】そして、「Build」ボタン701を選択することにより、上記したターゲットオブジェクトのIDL情報の取得がおこなわれ(ステップS603)、取得した情報が表示されて各データの編集を可能とした状態となる(ステップS604)。また、すでに復帰情報ファイルが存在している場合は、その復帰情報ファイル名を情報ファイル名入力フィールドに入力するとともに、「Edit」ボタン702を選択することで、その復帰情報ファイルの表示および編集が可能となる。また、図7中、「Back」703ボタンを選択することにより、図5に示したメニュー画面に戻ることもできる。

【0097】ここで、本発明にかかる分散オブジェクト開発システムの動作の理解を容易にするために、ターゲットオブジェクトのIDL定義を図8に示すような構成として、以下の説明をつづける。図8に示すIDLは、後述するクライアントシミュレータについての説明の際にも共通に用いるものとする。

【0098】なお、ステップS603におけるIDL情報の取得は、以下の手順にて自動的におこなわれる。まず、ネーミングサービスにアクセスして、上記したオブジェクト名のオブジェクト・リファレンスを取得する。つぎに、取得したオブジェクト・リファレンスを用いて、インプリメンテーションリポジトリにアクセスし、そのターゲットオブジェクトの実装状態を確認する。そして、上記したオブジェクト・リファレンスを用いてインタフェースリポジトリにアクセスし、そこからターゲットオブジェクトのIDL情報を取得する。

【0099】図9は、ステップS604において表示される情報ファイル(ここでは復帰情報を意味する)の編集画面を示す説明図である。図9に示すように、情報ファイル編集画面では、情報ファイル名表示フィールド910に、図7の情報ファイル名入力フィールド710に入力された情報ファイル名が表示され、また、オブジェクト名表示フィールド911に、図7のオブジェクト名入力フィールド720に入力されたオブジェクト名が表示される。また、インタフェース名表示フィールド912に、実際にネーミングサービスに登録されているターゲットオブジェクトの名前を示すインタフェース名が表示される。

【0100】そして、図8に示したIDLに従って、図6のステップS603において取得したIDL情報として、オペレーション名表示フィールド920に「op00」が表示され、戻り値属性表示フィールド921に「void」が表示される。また、同様に、パラメタ名表示フィールド930に「p2」が表示され、パラメタ属性表

10

20

30

40

50

示フィールド931に「long」が表示され、パラメタ値表示フィールド933に「-1」が表示される。

【0101】ここで、図示するように、記数方式選択部932によって、パラメタ値表示フィールド933に表示する記数方式を変更することができ、図9では10進記数方式を示す「Decimal」が選択されている。また、戻り値属性表示フィールド921に「long」等の戻り値を必要とする属性が表示される場合は、表示フィールド922に、記数方式選択部932およびパラメタ値表示フィールド933と同様な記数方式選択部および戻り値表示フィールドが表示される。

【0102】また、図9において、シミュレート繰り返し数入力フィールド940に、数値を入力することによって、後述するシミュレートをその入力した数値の示す回数だけ連続して繰り返すことができる。

【0103】そして、図9に示した情報ファイル編集画面の表示確認または編集を終えたい場合は、「Save」ボタン951を選択することにより、画面上に表示された内容が情報ファイル（この場合、復帰情報）として、情報ファイル名表示フィールド910に表示された情報ファイル名で保存される。すなわち、これにより、図6のステップS605において、復帰情報ファイルが作成される。

【0104】なお、復帰情報ファイルには、オブジェクト名、ネーミングサービス登録名、オペレーション名、戻り値名、戻り値属性、戻り値（戻り値属性に合わせて自動生成または編集される）、パラメタ名、パラメタ属性、およびパラメタ値（パラメタ属性に合わせて自動生成または編集される）が保存される。

【0105】また、図9において、「Simulator」ボタン952を選択することで、つづいてシミュレータ起動処理（ここでは、サーバシミュレータ）に移行することができ、また、「Back」ボタン953を選択することにより、図7に示した初期入力画面に戻ることもできる。

【0106】以上に説明したように、復帰情報作成処理によって、クライアントアプリケーションが呼び出すターゲットオブジェクトのIDL情報をORB上から動的に取得することができるので、アプリケーション開発者がこのIDL情報を従来のようにテキストイメージでのテストデータとしてあらかじめ作成する作業を必要としなくなる。また、取得したIDL情報を、オペレーション、名前、属性またはパラメタ毎にフィールドを設けてWWWブラウザ18上に表示しており、アプリケーション開発者による編集を容易にすることができる。

【0107】つづいて、図4に示すサーバシミュレータの動作の説明において、ステップS403の復帰情報作成処理後は、図9に示した「Simulator」ボタン952の選択、または図5に示した「Server Simulator」530の選択によって、サーバシミュレータが起動される（ステップS404）。

【0108】図10は、サーバシミュレータの初期入力画面を示す説明図であり、ステップS404におけるサーバシミュレータの起動指示によってWWWブラウザ18上に表示される画面を示している。ここで、サーバシミュレータの起動指示が、図9に示した「Simulator」ボタン952の選択によるものである場合は、復帰情報ファイルの作成が完了しているため、図10に示すように、その復帰情報ファイル名が、情報ファイル名入力フィールド1010に自動的に表示される。

【0109】また、同じ理由から、サーバ名表示フィールド1020に、図9のオブジェクト名表示フィールド911に表示されたオブジェクト名が、サーバ名として表示される。一方、サーバシミュレータの起動指示が、図5に示した「Server Simulator」530の選択によるものである場合は、それぞれ復帰情報ファイル名およびサーバ名を入力する必要がある。

【0110】さらに、サーバシミュレータの初期入力画面では、検証用ファイル名入力フィールド1030に、検証用ファイル名を入力する必要がある（ステップS405）。ここで、検証用ファイルとは、後述するように、起動情報および実行履歴を一括して保存したファイルであり、サーバシミュレータの実行終了後、この検証用ファイルの内容を調査することで、クライアントアプリケーションが正しく実行されたかどうかを検証することができる。

【0111】つづいて、「Execute」ボタン1041を選択することで、サーバシミュレータは、クライアントアプリケーションからの送信待ちの状態に移行する（ステップS406）。ステップS406においてクライアントアプリケーションから送信、すなわちターゲットオブジェクトの呼び出しリクエストが発生した場合（図3のステップS305に相当）は、復帰情報の動的編集の受け付け状態に移行する（ステップS407）。

【0112】復帰情報の動的編集とは、サーバシミュレータの起動後において動的に、上述した復帰情報作成処理をおこなうことであり、復帰情報の内容を、クライアントアプリケーションから送信されたリクエストに応じて動的に変更することができる。

【0113】ステップS407において動的編集の要求がない場合、または動的編集の処理が完了すると、保存された復帰情報ファイルが示す復帰情報を取り込み（ステップS408）、取り込んだ復帰情報をクライアントアプリケーションに返信する（ステップS409）。

【0114】そして、クライアントアプリケーションが送信したリクエストの内容（オブジェクト名、ネーミングサービス登録名、オペレーション名、パラメタ名、パラメタ値、区切りのキーワード）を起動情報として保存する（ステップS410）。この際、起動情報において、パラメタの型や値にIDLに従わないような誤りが存在する場合には、その旨を後述する実行履歴に出力す

る。

【0115】なお、この起動情報は、クライアントシミュレータの起動情報として再利用することができる。また、サーバシミュレータの実行状態（トレース状態）を実行履歴（実行日時、復帰情報ファイル名、ヘッダ、開始のキーワード、オブジェクト名、オペレーション名、サーバインタフェース動的実装状態、終了状態、終了のキーワード）として保存する（ステップS411）。

【0116】つぎに、アプリケーション開発者からの最新情報表示の要求をチェックする（ステップ412）。ここで、最新情報表示の要求とは、不定期にまたは連続的に実行されるクライアントアプリケーションの複数のリクエストに対して、上記した起動情報の表示および実行履歴の表示を直ちにおこなうことを指示することを意味し、図10における「View」ボタン1042の選択によって実現される。

【0117】ステップS412において、最新情報表示の要求が生じなかった場合は、後述するステップS416の終了要求のチェックをおこなう。また、ステップS412において、最新情報表示の要求が発生した場合は、図10における起動情報表示フィールド1060に、ステップS410において保存された起動情報の表示をおこなう（ステップS413）。図11は、この起動情報の表示形態を説明する説明図である。図11に示すように、起動情報は、サーバシミュレータの起動から停止に至るまでに受け取ったクライアントアプリケーションの複数のリクエスト毎に、その内容を起動情報表示フィールド1060に表示する。

【0118】なお、図中、例として、点線で囲まれた部分、すなわちリクエストAおよびBの内容が実際に起動情報表示フィールド1060に表示される部分であり、他のリクエストCの内容の表示は、起動情報表示フィールド1060をスクロール表示させることにより可能となる。

【0119】クライアントアプリケーションが送信するリクエストには、複数のターゲットオブジェクトの呼び出しを含めることが可能であるため、各ターゲットオブジェクトの呼び出しをそれぞれ独立したリクエストとすることもでき、この起動情報ファイルの単位でかつターゲットオブジェクト毎のリクエストを、起動情報表示フィールド1060の下方から上方に向けて表示していく。

【0120】図12は、上記した起動情報の表示例を示す説明図である。図12に示すように、起動情報表示フィールド1060には、オブジェクト名1202、ネーミングサービス登録名1203、オペレーション名1204、パラメタ名およびパラメタ値1205、区切りのキーワード1207が表示される。また、図示するように、見出し1201や復帰情報ファイル名1207を表示するようにしてもよい。復帰情報ファイル名1207

をマウスカーソル等でクリックすることにより、復帰情報作成処理へと移行し、図9に示した編集画面が表示されて、復帰情報ファイルの編集をおこなうようにすることもできる。

【0121】つぎに、ステップS413における起動情報の表示につづいて、図10における実行履歴表示フィールド1050に、ステップS411において保存された実行履歴の表示をおこなう（ステップS414）。図13は、この実行履歴の表示形態を説明する説明図である。図13に示すように、実行履歴は、図11に示した起動情報の表示に対応させて、サーバシミュレータの起動から停止に至るまでに受け取ったクライアントアプリケーションの複数のリクエスト毎に、その内容を実行履歴表示フィールド1050に表示する。

【0122】なお、図中、例として、点線で囲まれた部分、すなわちリクエストAおよびBの内容が実際に実行履歴表示フィールド1050に表示される部分であり、他のリクエストCの内容の表示は、実行履歴表示フィールド1050をスクロール表示させることにより可能となる。

【0123】また、図11に示したように、実行履歴ファイルの単位でかつターゲットオブジェクト毎のリクエストを、実行履歴表示フィールド1050の下方から上方に向けて表示していく。

【0124】図14は、上記した実行履歴の表示例を示す説明図である。図12に示すように、実行履歴表示フィールド1050には、実行日時1401、復帰情報ファイル名1402、ヘッダ（見出し）1403、開始のキーワード1404、オブジェクト名1405、オペレーション名1406、サーバインタフェース動的実装状態1407および1408、終了状態1409、終了のキーワード1410が表示される。

【0125】そして、ステップS414における実行履歴の表示につづき、検証用ファイルの作成をおこなう（ステップS415）。図15は、検証用ファイルの内部構成を説明する説明図である。図15に示すように、検証用ファイルは、ステップS411において作成された実行履歴の内容と、ステップS410において作成された起動情報の内容（ターゲットオブジェクト毎のリクエスト内容）を単位として、上述したように起動情報ファイルの単位毎にかつクライアントアプリケーションが送信したリクエスト単位毎に編成し、ステップS405において入力した検証用ファイル名で保存する。

【0126】すなわち、検証用ファイルは、サーバシミュレータの起動から停止に至るまでに受け取ったクライアントアプリケーションの複数のリクエスト毎に、図13に示した実行履歴と図11に示した起動情報とを順に対にして構成された内容を有して作成される。

【0127】そして、サーバシミュレータの実行を終了させる要求をチェックする（ステップS416）。な

お、この要求は、図10における「Quit」ボタン1043の選択によって実現される。ステップS416において、終了要求が発生しない場合は、再びステップS403の復帰情報作成処理に戻り、上述した処理を繰り返す。

【0128】また、ステップS416において、終了要求が生じた場合は、サーバシミュレータの動作を停止する。この場合、クライアントアプリケーション開発者は、開発したクライアントアプリケーションの実行をも終了させる。そして、最終的に、クライアントアプリケーション開発者は、上述した検証用ファイルを参照することにより、開発したクライアントアプリケーションの設計の誤りを検証する。

【0129】（サーバアプリケーションの開発手順）つぎに、図28に示した従来のサーバアプリケーション開発手順に対応させて、本発明にかかる分散オブジェクト開発システムにおけるクライアントシミュレータを用いたサーバアプリケーションの開発手順について説明する。図16は、本発明にかかる分散オブジェクト開発システムによるサーバアプリケーション開発手順を示すフローチャートである。図16に示すフローチャートは、図2と同様に、静的起動をおこなうサーバアプリケーションの開発をおこなう手順について示している。

【0130】図16において、まず、サーバアプリケーション開発者は、開発しようとするアプリケーションのプログラム・インタフェース設計をおこなう（ステップS1601）。つづいてサーバアプリケーションが備えた関数のIDLファイルを作成する（ステップS1602）。

【0131】そして、上記したIDLファイルを用いて実行される所望のサーバアプリケーションのソースを作成する（ステップS1603）。つぎに、ステップS1602において作成したIDLファイルを、IDLコンパイラを用いてコンパイルし、クライアントスタブ、サーバスケルトンおよびIDLオブジェクトを生成する（ステップS1604）。生成されたIDLオブジェクトは、インタフェースリポジトリに結合される（ステップS1605）。

【0132】つづいて、ステップS1603において作成されたサーバアプリケーションのソースをコンパイルしてオブジェクトを生成し、このオブジェクトに、ステップS1604において生成されたサーバスケルトンをリンクすることで、実行可能なサーバアプリケーションのファイルを得る（ステップS1606）。

【0133】ステップS1606において得られたサーバアプリケーションは、インプリメンテーションリポジトリに記録されるとともに、クライアントシミュレータによって呼び出し可能なオブジェクト・インプリメンテーションとして登録される（ステップS1607）。

【0134】そして、上記したクライアントシミュレー

タを起動する前処理を実行する（ステップS1608）。このクライアント起動前処理は、本発明にかかる分散オブジェクト開発システムによって実現される処理の一部に相当し、この処理の詳細については後述する。

【0135】その後、ステップS1606において実行可能なファイルとして生成されたサーバアプリケーションを実行させる（ステップS1609）。ここで実行されるサーバアプリケーションは、分散オブジェクト環境に実際に搭載する形態と同一のコードを有するものであり、特にテスト用のために変更を加える必要はない。すなわち、CORBA上で問題なく動作可能なバイナリソースのサーバアプリケーションを実行させる。

【0136】図17は、このサーバアプリケーションの実行処理を説明するためのフローチャートであり、特に静的起動をおこなうためのサーバスケルトンおよびオブジェクトアダプタがおこなう処理を示している。図17において、まずサーバアプリケーションは、初期化がおこなわれて、オブジェクトアダプタが、クライアントアプリケーション、特にクライアントスタブからオブジェクトの呼び出しリクエストを受け付ける（ステップS1701）。

【0137】受け取ったリクエストにはターゲットオブジェクトのオブジェクト・リファレンスが含まれており、オブジェクトアダプタは、このオブジェクト・リファレンスを用いて、インプリメンテーションリポジトリからターゲットオブジェクトの実装状態を取得する必要がある。

【0138】ただし、インプリメンテーションリポジトリもまたCORBA上のオブジェクトの一つであるため、あらかじめこのインプリメンテーションリポジトリのオブジェクト・リファレンスを取得する必要がある（ステップS1702）。これにより、インプリメンテーションリポジトリへのアクセスが可能となり、ターゲットオブジェクト（広義ではサーバ）の活性情報の取得が可能となる（ステップS1703）。

【0139】そして、オブジェクトアダプタは、ステップS1703において取得した活性情報から、サーバが活性済みであるか否かを判定する（ステップS1704）。ステップS1704においてサーバが活性されていない場合は、サーバを活性する（ステップS1705）。サーバの活性後またはステップS1704においてサーバが活性済みであった場合は、サーバスケルトンが、ターゲットオブジェクトの呼び出しをおこない（ステップS1706）、ターゲットオブジェクトから返信データを受け取るとともに、この返信データを発信元であるクライアントアプリケーションに送出する（ステップS1707）。

【0140】この返信データは、ORBコアを介して再び、クライアントスタブによってリクエスト発信元のクライアントアプリケーションに届けられるが、本発明に

10

20

30

40

50

かかる分散オブジェクト開発システムでは、この返信データ受け取り処理およびリクエスト送出処理を、上記したクライアントシミュレータがおこなう。よって、クライアントシミュレータは、換言すれば、クライアントスタブの機能をシミュレートすることにもなる。

【0141】ステップS1707の返信処理後、オブジェクトアダプタは、サーバの活性化をおこなった場合、そのサーバの非活性化等の後処理を実行する(ステップS1708)。

【0142】図16において、サーバアプリケーションを実行させた後は、クライアントシミュレータを起動する処理を実行する(ステップS1610)。このクライアントシミュレータ起動処理もまた、本発明にかかる分散オブジェクト開発システムによって実現される処理の一部に相当し、この処理の詳細についても後述する。

【0143】そして、クライアントシミュレータが送信したリクエストと、そのリクエストに対してサーバアプリケーションが返信した返信データと、によって、サーバアプリケーションの動作結果の検証がおこなわれる(ステップS1611)。

【0144】(クライアントシミュレータの動作) つぎに、クライアントシミュレータの動作について説明する。このクライアントシミュレータの動作は、上述したクライアントシミュレータ起動前処理およびクライアントシミュレータ起動処理に相当するものである。図18は、本発明にかかる分散オブジェクト開発システムのクライアントシミュレータの動作を示すフローチャートである。特に、ここでは、WWWブラウザ18上でクライアントシミュレータの制御をおこなう場合について説明する。

【0145】図18において、まず、アプリケーション開発者は、分散オブジェクト環境12の下にあるノードの一つを使用して、WWWブラウザ18を起動させる。そして、図5に示したメニュープログラムを起動させる(ステップS1801)。そして、クライアントシミュレータ用の情報ファイル、すなわち起動情報を作成するため、図5に示した画面上において、「Simulation Data Editor」510を選択する。なお、情報ファイルの作成処理は、上述したサーバシミュレータの説明においておこなったものと同様であるので、ここではその説明を省略する。

【0146】よって、上述したような選択手段により、クライアントシミュレータモードの選択をおこなう(ステップS1802)。この選択により、起動情報作成処理へと処理が移行する(ステップS1803)。ここで、起動情報とは、上記した情報ファイルをクライアントシミュレータモードにおいて作成する場合のその情報ファイルの呼び名である。具体的には、この起動情報は、クライアントからサーバへ送信されるリクエストを示す。

【0147】(起動情報作成処理) つぎに、ステップS1803における起動情報作成処理について説明する。なお、この起動情報作成処理は、図16のステップS1608におけるクライアントシミュレータ起動前処理に相当する処理でもある。図19は、ステップS1803におけるクライアントシミュレータの起動情報作成処理を示すフローチャートである。なお、情報ファイル(ここでは起動情報を意味する)作成処理における初期入力画面については図7に示した図と同様であるため、これを利用する。

【0148】図7において、まず、情報ファイル名入力フィールド710に、起動情報ファイル名を入力する(ステップS1901)。この起動情報ファイル名とは、これから作成する起動情報の格納先となるファイルの名前であり、たとえば「/home/apts/dat/apt0001_cl.dat」のようなものである。また、この起動情報ファイル名は、後述するクライアントシミュレータの起動時において参照される。

【0149】つづいて、オブジェクト名入力フィールド720に、サーバオブジェクト名を入力する(ステップS1902)。このオブジェクト名とは、サーバアプリケーションが受け付け、かつクライアントシミュレータ上で送信されるリクエストに含まれるターゲットオブジェクトの名前であり、たとえば「IDL:apts0001/apts0001:1.0」のようなものである。

【0150】そして、「Build」ボタン701を選択することにより、上記したターゲットオブジェクトのIDL情報の取得がおこなわれ(ステップS1903)、取得した情報が表示されて各データの編集を可能とした状態となる(ステップS1904)。また、すでに起動情報ファイルが存在している場合は、その起動情報ファイル名を情報ファイル名入力フィールドに入力するとともに、「Edit」ボタン702を選択することで、その起動情報ファイルの表示および編集が可能となる。

【0151】ステップS1903におけるIDL情報の取得は、図6のステップS603と同様であり、ステップS1904において表示される情報ファイル(ここでは起動情報を意味する)の編集画面もまた、図6のステップS604と同様であるので、ここではこれらの説明を省略する。

【0152】よって、この起動情報作成処理により、オブジェクト名、ネーミングサービス登録名、オペレーション名、戻り値名、戻り値属性、パラメタ名、パラメタ属性およびパラメタ値が保存された起動情報ファイルが作成される(ステップS1905)。

【0153】なお、図9に示した「Simulator」ボタン952を選択することで、つづいてシミュレータ起動処理(ここでは、クライアントシミュレータ)に移行することができる。

【0154】以上に説明したように、起動情報作成処理

によって、サーバアプリケーションが受け付けるターゲットオブジェクトのIDL情報をORB上から動的に取得することができるので、アプリケーション開発者がこのIDL情報を従来のようにテキストイメージでのテキストデータとしてあらかじめ作成する作業を必要としなくなる。また、取得したIDL情報を、オペレーション、名前、属性またはパラメタ毎にフィールドを設けてWWWブラウザ18上に表示しており、アプリケーション開発者による編集を容易にすることができる。

【0155】 つづいて、図18に示すサーバシミュレータの動作の説明において、ステップS1803の復帰情報作成処理後は、図9に示した「Simulator」ボタン952の選択、または図5に示した「Client Simulator」520の選択によって、クライアントシミュレータが起動される(ステップS1804)。

【0156】 図20は、クライアントシミュレータの初期入力画面を示す説明図であり、ステップS1804におけるクライアントシミュレータの起動指示によってWWWブラウザ18上に表示される画面を示している。ここで、クライアントシミュレータの起動指示が、図9に示した「Simulator」ボタン952の選択によるものである場合は、起動情報ファイルの作成が完了しているため、図20に示すように、その起動情報ファイル名が、情報ファイル名入力フィールド2010に自動的に表示される。

【0157】 一方、サーバシミュレータの起動指示が、図5に示した「Client Simulator」520の選択によるものである場合は、起動情報ファイル名を入力する必要がある。

【0158】 さらに、クライアントシミュレータの初期入力画面では、検証用ファイル名入力フィールド2020に、検証用ファイル名を入力する必要がある(ステップS1805)。ここで、検証用ファイルとは、上述したように、起動情報および実行履歴を一括して保存したファイルであり、クライアントシミュレータの実行終了後、この検証用ファイルの内容を調査することで、サーバアプリケーションが正しく実行されたかどうかを検証することができる。

【0159】 つづいて、「Try」ボタン2041を選択することで、クライアントシミュレータは、保存された起動情報ファイルが示す起動情報を取り込み(ステップS1806)、取り込んだ起動情報をリクエストとしてサーバアプリケーションに送信する(ステップS1807)。

【0160】 そして、サーバアプリケーションからの返信待ちの状態に移行する(ステップS1808)。ステップS1808においてサーバアプリケーションから返信があった場合(図17のステップS1707に相当)は、その返信の内容(オブジェクト名、ネーミングサービス登録名、オペレーション名、パラメタ名、パラメタ

値、区切りのキーワード)を復帰情報として保存する(ステップS1809)。この際、復帰情報において、パラメタの型や値にIDLに従わないような誤りが存在する場合には、その旨を後述する実行履歴に出力する。

【0161】 なお、この復帰情報ファイルは、サーバシミュレータの復帰情報として再利用することができる。また、クライアントシミュレータの実行状態(トレース状態)を実行履歴(実行日時、復帰情報ファイル名、ヘッダ、開始のキーワード、オブジェクト名、オペレーション名、サーバインタフェース動的実装状態、リクエスト送信、返信受信、終了状態、終了のキーワード)として保存する(ステップS1810)。

【0162】 そして、図20における起動情報表示フィールド2060に、ステップS1809において保存された復帰情報の表示をおこなう(ステップS1811)。図21は、この復帰情報の表示形態を説明する説明図である。図21に示すように、復帰情報は、クライアントシミュレータの起動から停止に至るまでに受け取ったサーバアプリケーションの複数の返信毎に、その内容を復帰情報表示フィールド2060に表示する。

【0163】 なお、図中、例として、点線で囲まれた部分の内容が実際に復帰情報表示フィールド2060に表示される部分であり、他の内容の表示は、復帰情報表示フィールド2060をスクロール表示させることにより可能となる。この返信内容の表示は、サーバアプリケーションからの返信を受け取る毎に、その返信内容を復帰情報として復帰情報表示フィールド2060の下方向から上方に向けて表示していく。

【0164】 図22は、上記した復帰情報の表示例を示す説明図である。図22に示すように、復帰情報表示フィールド2060には、オブジェクト名2201、ネーミングサービス登録名2202、オペレーション名2203、パラメタ名およびパラメタ値2204、区切りのキーワード2205が表示される。

【0165】 つぎに、ステップS1811における復帰情報の表示につづいて、図20における実行履歴表示フィールド2050に、ステップS1810において保存された実行履歴の表示をおこなう(ステップS1812)。図23は、この実行履歴の表示形態を説明する説明図である。図23に示すように、実行履歴は、図21に示した復帰情報の表示に対応させて、クライアントシミュレータの起動から停止に至るまでに受け取ったサーバアプリケーションの返信毎に、その内容を実行履歴表示フィールド2050に表示する。

【0166】 なお、図中、例として、点線で囲まれた部分の内容が実際に実行履歴表示フィールド2050に表示される部分であり、他の返信内容の表示は、実行履歴表示フィールド2050をスクロール表示させることにより可能となる。また、図21に示したように、この実行履歴は、クライアントシミュレータのリクエスト送信

毎に、実行履歴表示フィールド2050の下方から上方に向けて表示していく。

【0167】図24は、上記した実行履歴の表示例を示す説明図である。図24に示すように、実行履歴表示フィールド2050には、実行日時2301、起動情報ファイル名2302、ヘッダ（見出し）2303、開始のキーワード2304、オペレーション名2305、サーバインタフェース動的実装状態2306および2307、リクエスト送信2308、返信受信2309、終了状態2310、終了のキーワード2311が表示される。

【0168】そして、ステップS1812における実行履歴の表示につづき、検証用ファイルの作成をおこなう（ステップS1813）。図25は、検証用ファイルの内部構成を説明する説明図である。図25に示すように、検証用ファイルは、ステップS1810において作成された実行履歴の内容と、ステップS1809において作成された復帰情報の内容（サーバアプリケーションからの返信内容）を単位として編成し、ステップS1805において入力した検証用ファイル名で保存する。

【0169】すなわち、検証用ファイルは、クライアントシミュレータの起動から停止に至るまでに受け取ったサーバアプリケーションの返信毎に、図23に示した実行履歴と図21に示した起動情報とを順に対にして構成された内容を有して作成される。

【0170】そして、クライアントシミュレータの実行を終了させる要求をチェックする（ステップS1814）。なお、この要求は、図20における「Quit」ボタン2042の選択によって実現される。ステップS1814において、終了要求が発生しない場合は、再びステップS1803の起動情報作成処理に戻り、上述した処理を繰り返す。

【0171】また、ステップS1814において、終了要求が生じた場合は、クライアントシミュレータの動作を停止する。この場合、サーバアプリケーション開発者は、開発したサーバアプリケーションの実行をも終了させる。そして、最終的に、サーバアプリケーション開発者は、上述した検証用ファイルを参照することにより、開発したサーバアプリケーションの設計の誤りを検証する。

【0172】以上に説明したように、実施の形態にかかる分散オブジェクト開発システムおよび分散オブジェクト開発方法によれば、IDL情報を取得することで、クライアントアプリケーションの実行検証をおこなう際にテスト用サーバアプリケーションを必要とする場合やサーバアプリケーションの実行検証をおこなう際にテスト用クライアントアプリケーションを必要とする場合に準備されるテストデータ（情報ファイル）を、取得したIDL情報を利用して自動的に作成するので、このようなテストデータの作成を手作業でおこなうことなく、アプ

リケーション開発者の負担を軽減させることができる。

【0173】また、このように作成したテストデータを用いてオブジェクトの呼び出しおよび／またはオブジェクトの実行をおこなうことにより、サーバアプリケーションまたはクライアントアプリケーションをシミュレートすることができるので、テスト用アプリケーションの開発を必要とせず、アプリケーション開発に要する時間を短縮させることが可能となる。

10 【0174】WWWサーバ20上においてサーバシミュレータまたはクライアントシミュレータを自動的に構築することで、クライアントおよびサーバアプリケーションの開発環境を分離するとともに、WWWブラウザ18上を介して、サーバシミュレータまたはクライアントシミュレータの動作を制御できるので、所望のホストシステム上にアプリケーション開発環境を構築することができる。

【0175】なお、以上に説明した実施の形態において、図4および図18に示した処理を実行させるシミュレートプログラムを、ICカードメモリ、フロッピーディスク、光磁気ディスク、CD-ROM等の記録媒体に格納し、この記録媒体に記録されたプログラムを、分散オブジェクト環境12上のいずれかのノード（シミュレータマシン10およびWWWサーバ20を含む）にインストールすることで、上述した分散オブジェクト開発方法を使用し、これにより分散オブジェクト開発システムを構築することもできる。なお、このインストール作業は、通信回線を使用して一方のノードが他方のノードからダウンロードすることによっておこなってもよい。

【0176】

30 【発明の効果】以上、説明したとおり、本発明にかかる分散オブジェクト開発システム（請求項1）、および、分散オブジェクト開発をコンピュータに実行させるためのプログラムを記録したコンピュータ読み取り可能な記録媒体（請求項9）によれば、分散オブジェクト環境において、オブジェクトの型情報を定義するとともにクライアントアプリケーションとサーバアプリケーションとの通信を仲介するインタフェース定義情報を取得し、取得した情報に基づいて、オブジェクトの呼び出しおよび／またはオブジェクトの実行をシミュレートするので、アプリケーションの実行検証をおこなう際に対となるテスト用のアプリケーションの実行に必要なテストデータを、従来のように手作業で作成することなく、自動的に生成させて、アプリケーション開発者の負担を軽減させることができ、このテストデータをオブジェクトの呼び出しおよび／またはオブジェクトの実行をおこなう際の通信電文として用いることでサーバシミュレータまたはクライアントシミュレータを自動的に構築することができ、アプリケーション開発に要する時間を短縮させることが可能となるという効果を奏する。

50 【0177】また、本発明にかかる分散オブジェクト開

発システム（請求項2）、および、分散オブジェクト開発をコンピュータに実行させるためのプログラムを記録したコンピュータ読み取り可能な記録媒体（請求項10）によれば、前記オブジェクトの型情報を定義するとともに前記クライアントアプリケーションと前記サーバアプリケーションとの通信を仲介するインタフェース定義情報を取得し、取得した情報に基づいて、前記クライアントアプリケーションによる前記オブジェクトの呼び出し要求に応じた応答を生成し、生成された応答を前記クライアントアプリケーションに対して送信し、また、前記インタフェース定義情報を取得し、取得した情報に基づいて、前記オブジェクトの呼び出し要求を生成し、生成された要求を前記サーバアプリケーションに対して送信して、該要求に応じて前記サーバアプリケーションから送信された応答を受信するので、サーバアプリケーションおよびクライアントアプリケーションをシミュレートする機能を自動的に構築することができ、アプリケーション開発者の負担の軽減およびアプリケーション開発に要する時間の短縮を図ることができるという効果を奏する。

【0178】また、本発明にかかる分散オブジェクト開発システム（請求項3）によれば、複数のノードのうちの少なくとも一つのノードが前記サーバアプリケーションシミュレート手段および／または前記クライアントアプリケーションシミュレート手段を備えており、当該ノードを除く他のノードのうちの所望のノードが、前記ネットワークを介して、前記サーバアプリケーションシミュレート手段および／または前記クライアントアプリケーションシミュレート手段のシミュレート機能を制御するので、テストの対象となるアプリケーションと、シミュレートされるアプリケーションとをネットワーク上に分離して配置することができるという効果を奏する。

【0179】また、本発明にかかる分散オブジェクト開発システム（請求項4）によれば、前記サーバアプリケーションシミュレート手段および／または前記クライアントアプリケーションシミュレート手段のシミュレート機能を、前記ノードの一つであるWWWサーバ上に備え、前記ノードの他の一つがWWWブラウザを介して前記シミュレート機能を制御するので、アプリケーション開発者が、WWWブラウザ上のGUI表示による容易な操作を享受することができ、アプリケーション開発の効率を高めることができるとともに、WWWブラウザを介して、所望のホストシステム上にアプリケーション開発環境を構築することができるという効果を奏する。

【0180】また、本発明にかかる分散オブジェクト開発システム（請求項5）によれば、前記サーバアプリケーションシミュレート手段が、前記クライアントアプリケーションによる前記オブジェクトの呼び出し要求の内容と、前記生成された応答の内容と、から構成される検証用ファイルを作成するので、この検証用ファイルを参

照することにより、開発したクライアントアプリケーションの設計の誤りを検証することができるという効果を奏する。

【0181】また、本発明にかかる分散オブジェクト開発システム（請求項6）によれば、前記サーバアプリケーションシミュレート手段が、前記クライアントアプリケーションによる前記オブジェクトの呼び出し要求の内容が前記インタフェースの定義に従った型と一致しない場合に、型の不一致を示す情報を前記生成された応答の内容に付加して、前記検証用ファイルを作成するので、検証用ファイルを参照することで、開発したクライアントアプリケーションの設計の誤りを容易に発見することができるという効果を奏する。

【0182】また、本発明にかかる分散オブジェクト開発システム（請求項7）によれば、前記クライアントアプリケーションシミュレート手段が、前記生成された要求の内容と、該要求に応じて前記サーバアプリケーションから送信された応答の内容と、から構成される検証用ファイルを作成するので、この検証用ファイルを参照することにより、開発したサーバアプリケーションの設計の誤りを検証することができるという効果を奏する。

【0183】また、本発明にかかる分散オブジェクト開発システム（請求項8）によれば、前記クライアントアプリケーションシミュレート手段が、前記サーバアプリケーションから返信された応答の内容が前記インタフェースの定義に従った型と一致しない場合に、型の不一致を示す情報を前記生成された要求の内容に付加して、前記検証用ファイルを作成するので、開発したサーバアプリケーションの設計の誤りを容易に発見することができるという効果を奏する。

【図面の簡単な説明】

【図1】本発明にかかる分散オブジェクト開発システムの概略構成を示すブロック図である。

【図2】本発明にかかる分散オブジェクト開発システムによるクライアントアプリケーション開発手順を示すフローチャートである。

【図3】クライアントアプリケーションの実行処理を説明するためのフローチャートである。

【図4】本発明にかかる分散オブジェクト開発システムのサーバシミュレータの動作を示すフローチャートである。

【図5】本発明にかかる分散オブジェクト開発システムの起動メニュー画面を示す説明図である。

【図6】本発明にかかる分散オブジェクト開発システムのサーバシミュレータの復帰情報作成処理を示すフローチャートである。

【図7】本発明にかかる分散オブジェクト開発システムの情報ファイル作成処理における初期入力画面を示す説明図である。

【図8】本発明にかかる分散オブジェクト開発システム

の説明に用いるIDLを示す図である。

【図9】本発明にかかる分散オブジェクト開発システムの情報ファイルの編集画面を示す説明図である。

【図10】本発明にかかる分散オブジェクト開発システムのサーバシミュレータの初期入力画面を示す説明図である。

【図11】本発明にかかる分散オブジェクト開発システムのサーバシミュレータにおける起動情報の表示形態を説明する説明図である。

【図12】本発明にかかる分散オブジェクト開発システムのサーバシミュレータにおける起動情報の表示例を示す説明図である。

【図13】本発明にかかる分散オブジェクト開発システムのサーバシミュレータにおける実行履歴の表示形態を説明する説明図である。

【図14】本発明にかかる分散オブジェクト開発システムのサーバシミュレータにおける実行履歴の表示例を示す説明図である。

【図15】本発明にかかる分散オブジェクト開発システムのサーバシミュレータにおける検証用ファイルの内部構成を説明する説明図である。

【図16】本発明にかかる分散オブジェクト開発システムによるサーバアプリケーション開発手順を示すフローチャートである。

【図17】サーバアプリケーションの実行処理を説明するためのフローチャートである。

【図18】本発明にかかる分散オブジェクト開発システムのクライアントシミュレータの動作を示すフローチャートである。

【図19】本発明にかかる分散オブジェクト開発システムのクライアントシミュレータの起動情報作成処理を示すフローチャートである。

【図20】本発明にかかる分散オブジェクト開発システムのクライアントシミュレータの初期入力画面を示す説*

* 明図である。

【図21】本発明にかかる分散オブジェクト開発システムのクライアントシミュレータにおける復帰情報の表示形態を説明する説明図である。

【図22】本発明にかかる分散オブジェクト開発システムのクライアントシミュレータにおける復帰情報の表示例を示す説明図である。

【図23】本発明にかかる分散オブジェクト開発システムのクライアントシミュレータにおける実行履歴の表示形態を説明する説明図である。

【図24】本発明にかかる分散オブジェクト開発システムのクライアントシミュレータにおける実行履歴の表示例を示す説明図である。

【図25】本発明にかかる分散オブジェクト開発システムのクライアントシミュレータにおける検証用ファイルの内部構成を説明する説明図である。

【図26】CORBAにおけるORBのインタフェース構造を示すブロック図である。

【図27】従来におけるクライアントアプリケーション開発手順を示すフローチャートである。

【図28】従来におけるサーバアプリケーション開発手順を示すフローチャートである。

【符号の説明】

10 シミュレータマシン

10a サーバアプリケーションシミュレーション部

10b クライアントアプリケーションシミュレーション部

10c プログラムインタフェース実装部

12 分散オブジェクト環境

14 ネットワーク

16 IDL定義格納部

18 WWWブラウザ

20 WWWサーバ

22 コマンド部

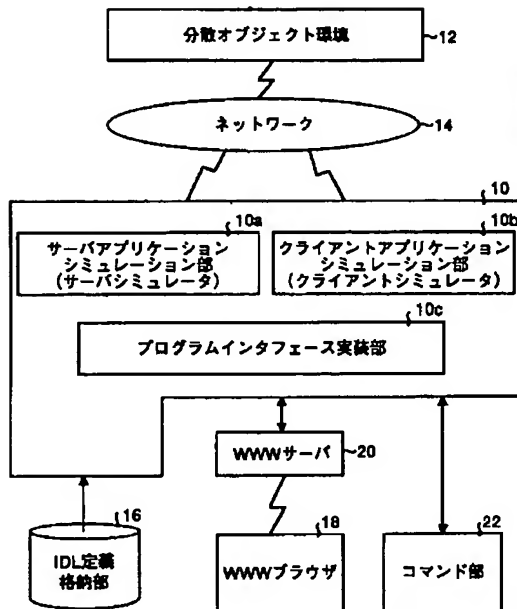
【図8】

本発明にかかる分散オブジェクト開発システムの説明に用いるIDLを示す図

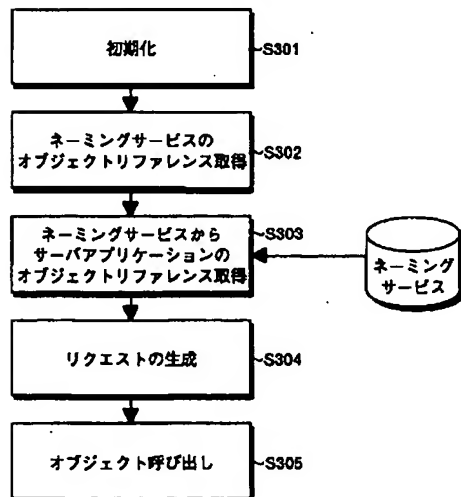
```
module      apts0001 {
    interface apts0001 {
        void op00 ( in long p1 , out long p2 );
    };
};
```

【図1】

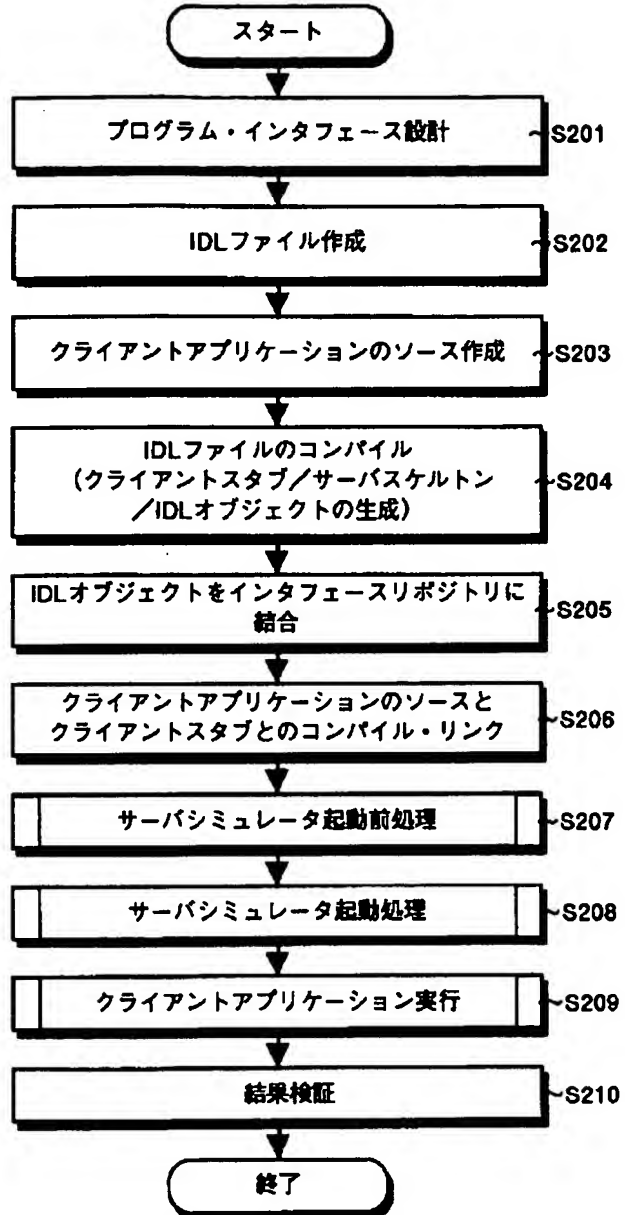
本発明にかかる分散オブジェクト開発システムの概略構成図



【図3】

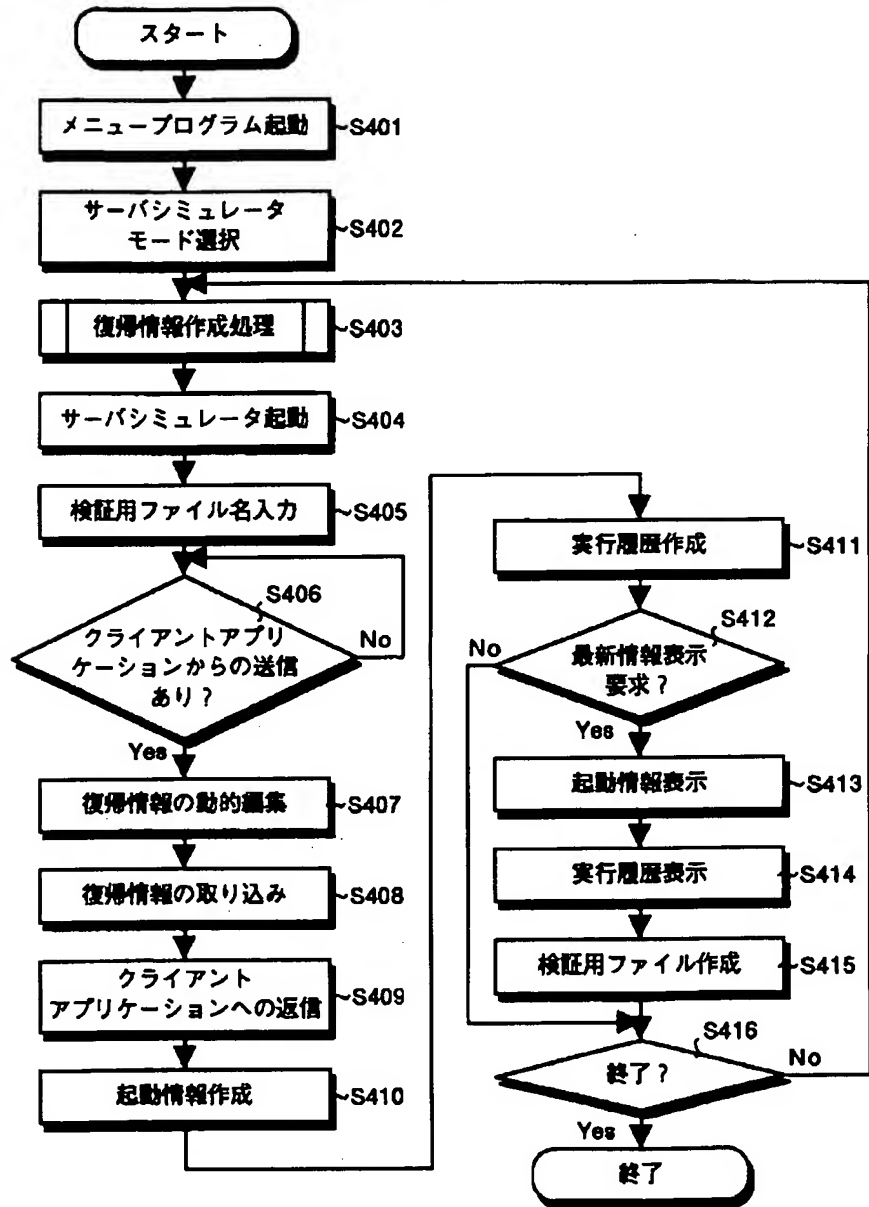
クライアントアプリケーションの実行処理を
説明するためのフローチャート

【図2】

本発明にかかる分散オブジェクト開発システムによる
クライアントアプリケーション開発手順を示すフローチャート

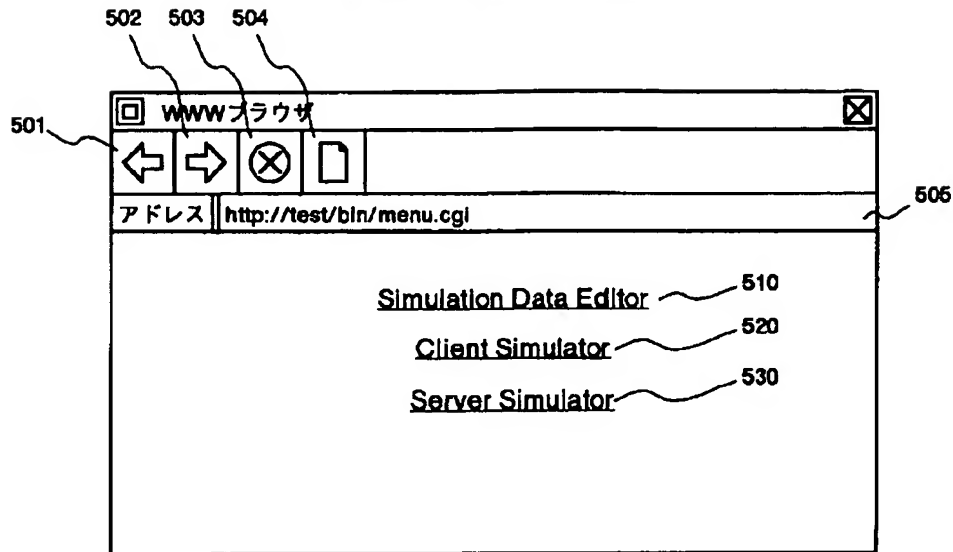
【図4】

本発明にかかる分散オブジェクト開発システムの
サーバシミュレータの動作を示すフローチャート



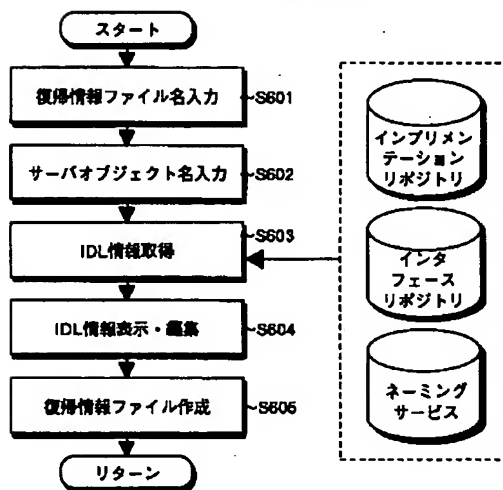
【図5】

起動メニュー画面を示す説明図



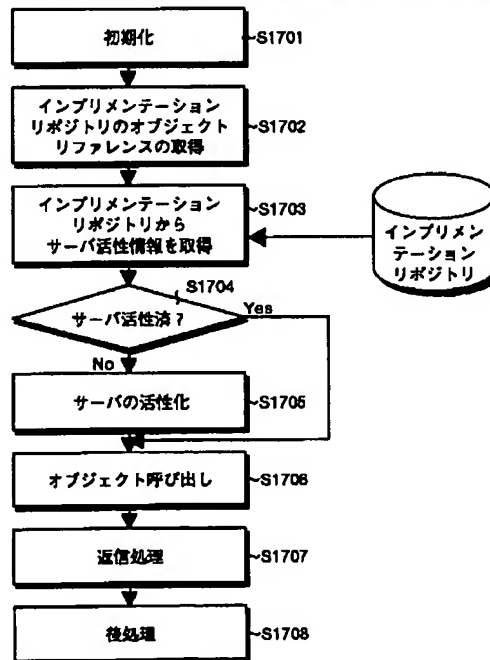
【図6】

サーバシミュレータの復帰情報作成処理を示すフローチャート



【図17】

サーバアプリケーションの実行処理を説明するためのフローチャート



【図7】

情報ファイル作成処理における初期入力画面を示す説明図

WWWブラウザ

アドレス http://

Simulation Data Editor

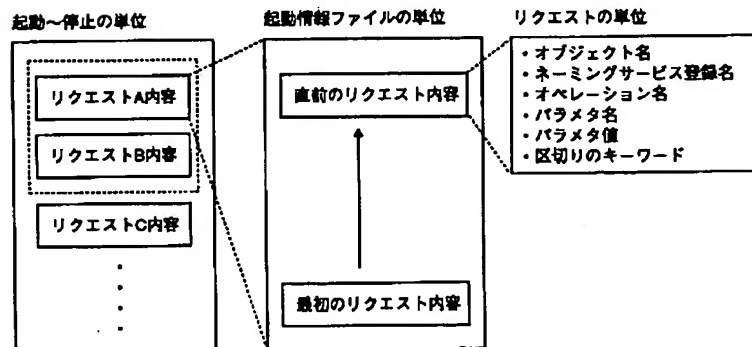
Simulation Data File Name: 710

Object Name: 720

Build 701 Edit 702 Back 703

【図11】

サーバシミュレータにおける起動情報の表示形態を説明する説明図



【図9】

情報ファイルの編集画面を示す説明図

wwwブラウザ

アドレス http://

Simulation Data Editor

Simulation Data File Name	/home/apts/dat/apt0001_sv.dat		
Object Name	IDL:apts0001/apts0001:1.0		
Interface Name	apts0001::apts0001		

RESULT	operation	op00	
Type	void		

OUT	para	p2	
Type	long	Decimal ▼	value -1

INVOKE COUNT

Save Simulator Back

【図12】

サーバシミュレータにおける起動情報の表示例を示す説明図

```

1201 *****/home/apts/dat/apts0001_sv.dat  REQUEST FILE start *****
1202 ~replid IDL:apts0001/apts0001:1.0
1203 ~Interface apts0001::apts0001
1204 ~operation op00
1205 ~para p1 =2147483847
1206 ~Invoke
1207 ~/home/apts/dat/apts0001_sv.dat

```

【図22】

クライアントシミュレータにおける復帰情報の表示例を示す説明図

```

2201 ~replid IDL:apts0001/apts0001:1.0
2202 ~Interface apts0001::apts0001
2203 ~operation op00
2204 ~para p2 = -1
2205 ~Invoke

```

【図10】

サーバシミュレータの初期入力画面を示す説明図

wwwブラウザ

← → × □

アドレス http://

Server Simulator

Simulation Data File Name: /home/aps/dat/aps0001_sv.dat 1010

Server Name: IDL:aps0001/aps0001:1.0 1020

History Log File Name: 1030

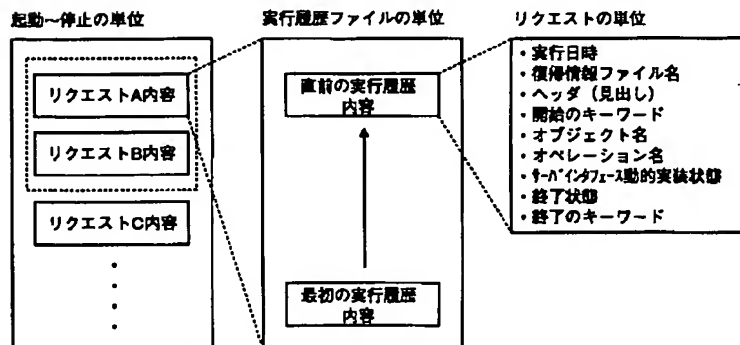
Execute 1041 View 1042 Quit 1043

1050

1060

【図13】

サーバシミュレータにおける実行履歴の表示形態を説明する説明図



【図14】

サーバシミュレータにおける実行履歴の表示例を示す説明図

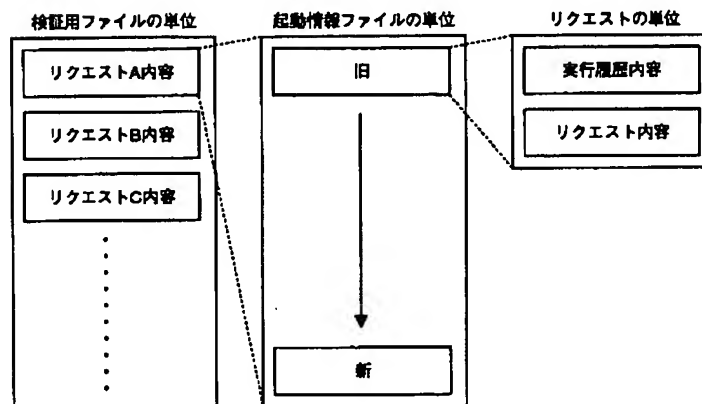
```

1401 CREATE DATE : Mon Jan 1 13:44:54 1999
1402 INVOKE FILE : /home/apts/dst/apts0001_sv.dat
1403 SEQ TIME KIND MESSAGE
-----
1404 1 13:44:54.804 SERVER ***** SERVER TP INVOKE START *****
1405 2 13:44:54.804 SERVER REPID = IDL:apts0001/apts0001:1.0
1406 3 13:44:54.804 SERVER OPERATION = op00
1407 4 13:44:54.811 SERVER CORBA_NVLst_add_item = p1
1408 5 13:44:54.811 SERVER CORBA_NVLst_add_item = p2
1409 6 13:44:54.819 SERVER Nomal End
1410 7 13:44:54.819 SERVER ***** SERVER TP INVOKE END *****

```

【図15】

サーバシミュレータにおける検証用ファイルの内部構成を説明する説明図

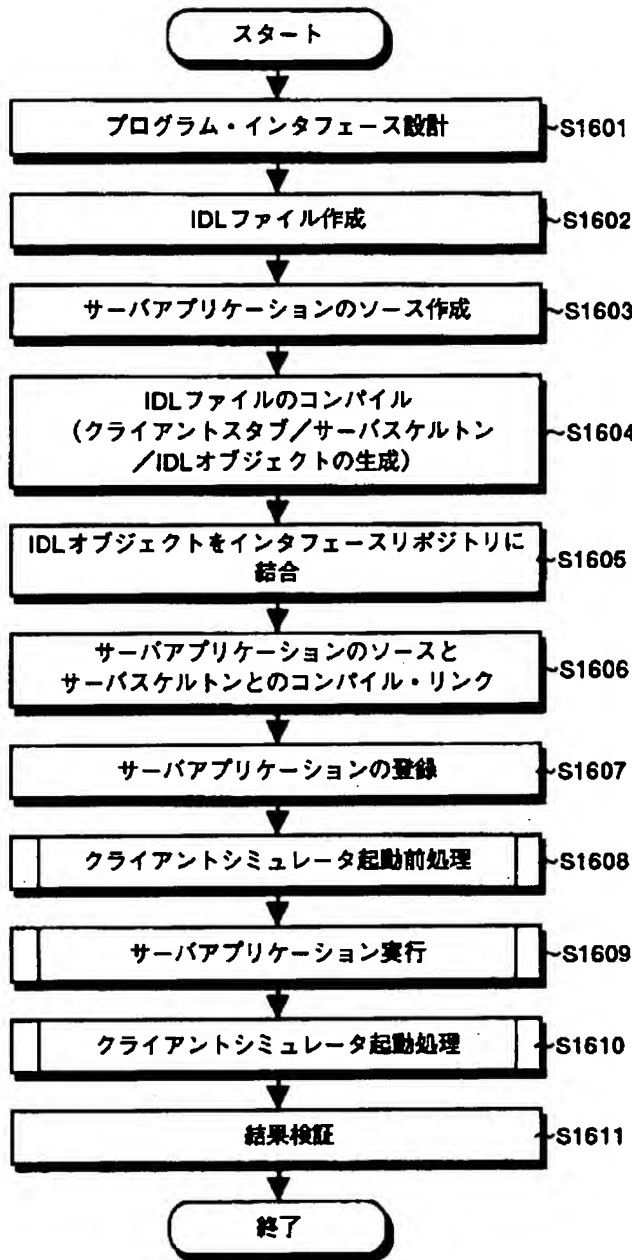


【図20】

クライアントシミュレータの初期入力画面を示す説明図

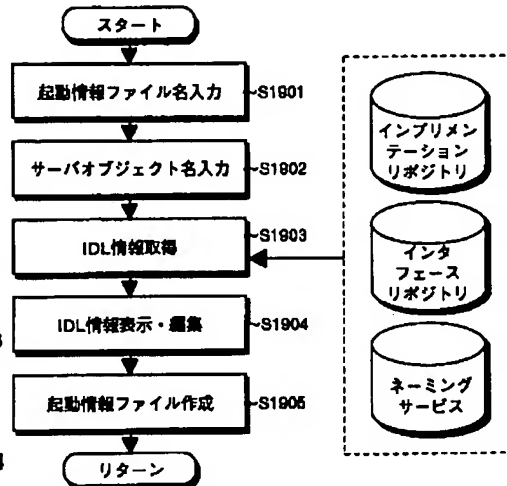
【図16】

本発明にかかる分散オブジェクト開発システムによる
サーバアプリケーション開発手順を示すフローチャート



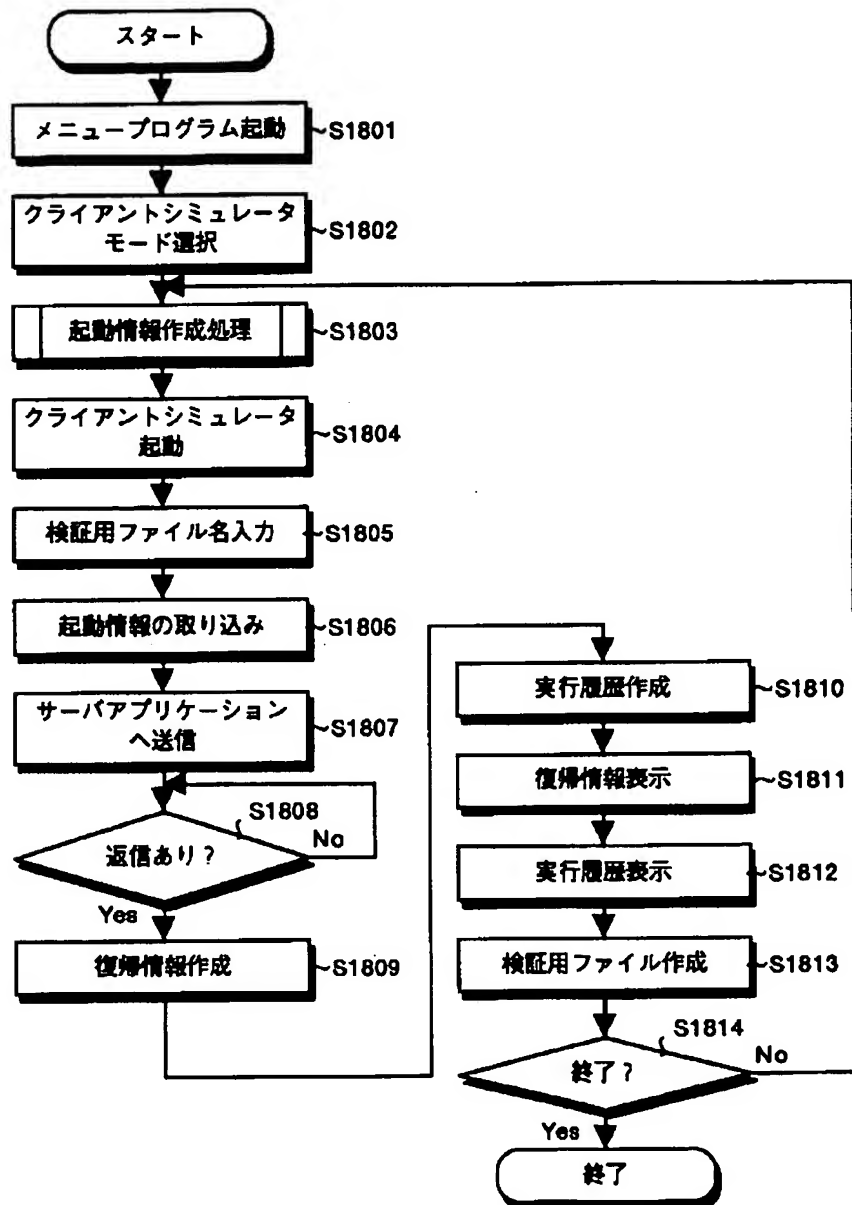
【図19】

クライアントシミュレータの起動情報作成処理を示すフローチャート



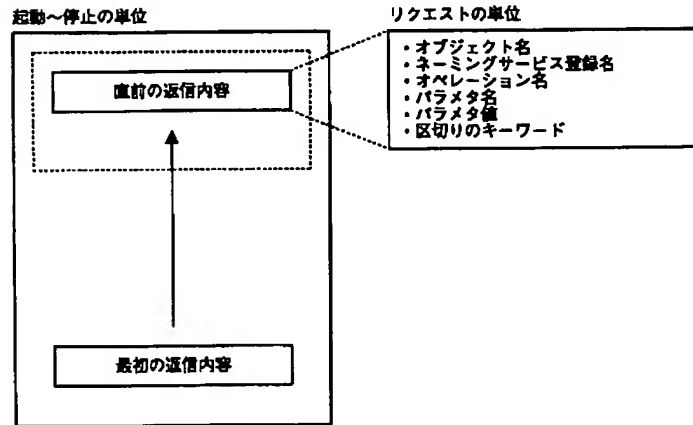
【図18】

本発明にかかる分散オブジェクト開発システムの
クライアントシミュレータの動作を示すフローチャート



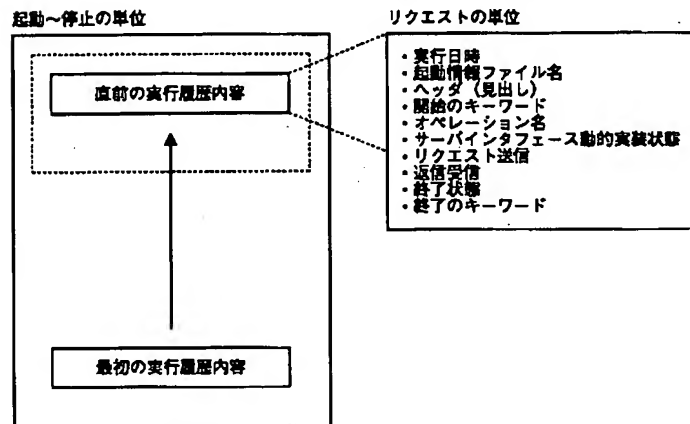
【図21】

クライアントシミュレータにおける復帰情報の表示形態を説明する説明図



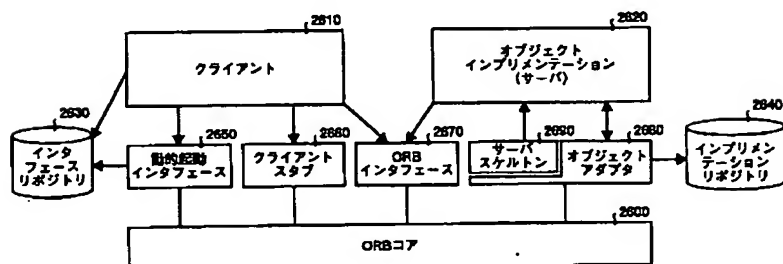
【図23】

クライアントシミュレータにおける実行履歴の表示形態を説明する説明図



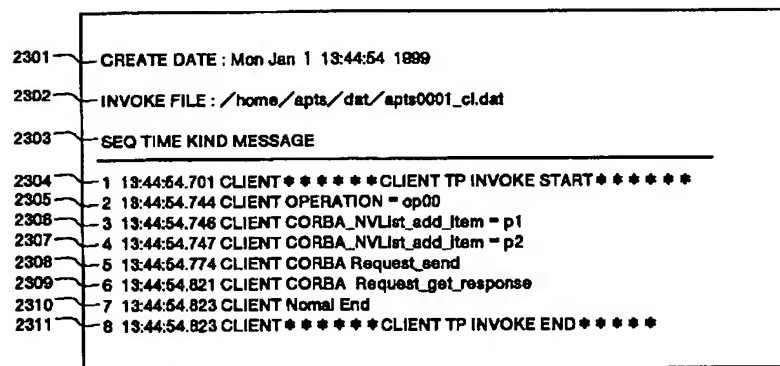
【図26】

CORBAにおけるORBのインタフェース構造を示すブロック図



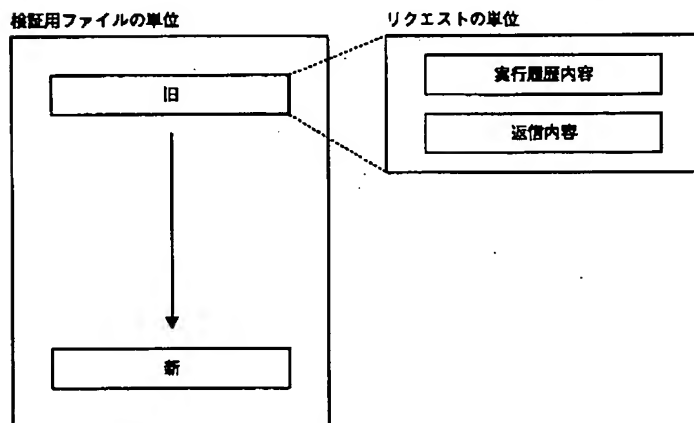
【図24】

クライアントシミュレータにおける実行履歴の表示例を示す説明図



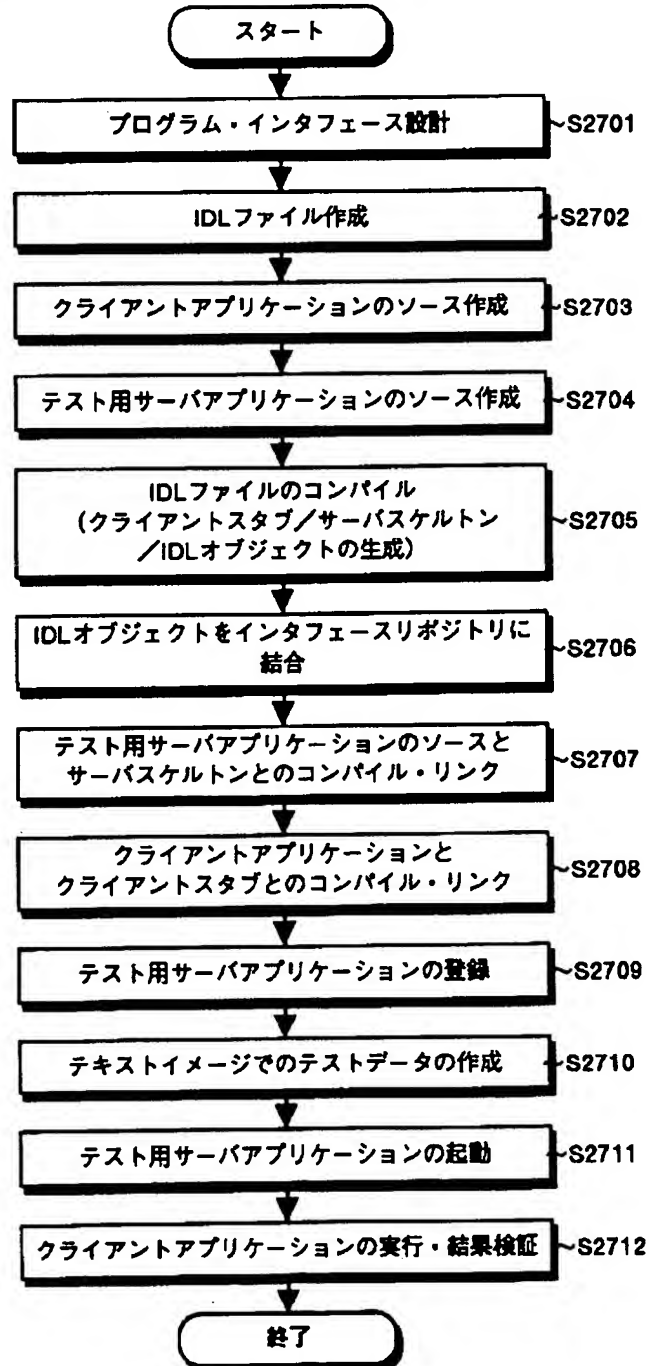
【図25】

クライアントシミュレータにおける検証用ファイルの内部構成を説明する説明図



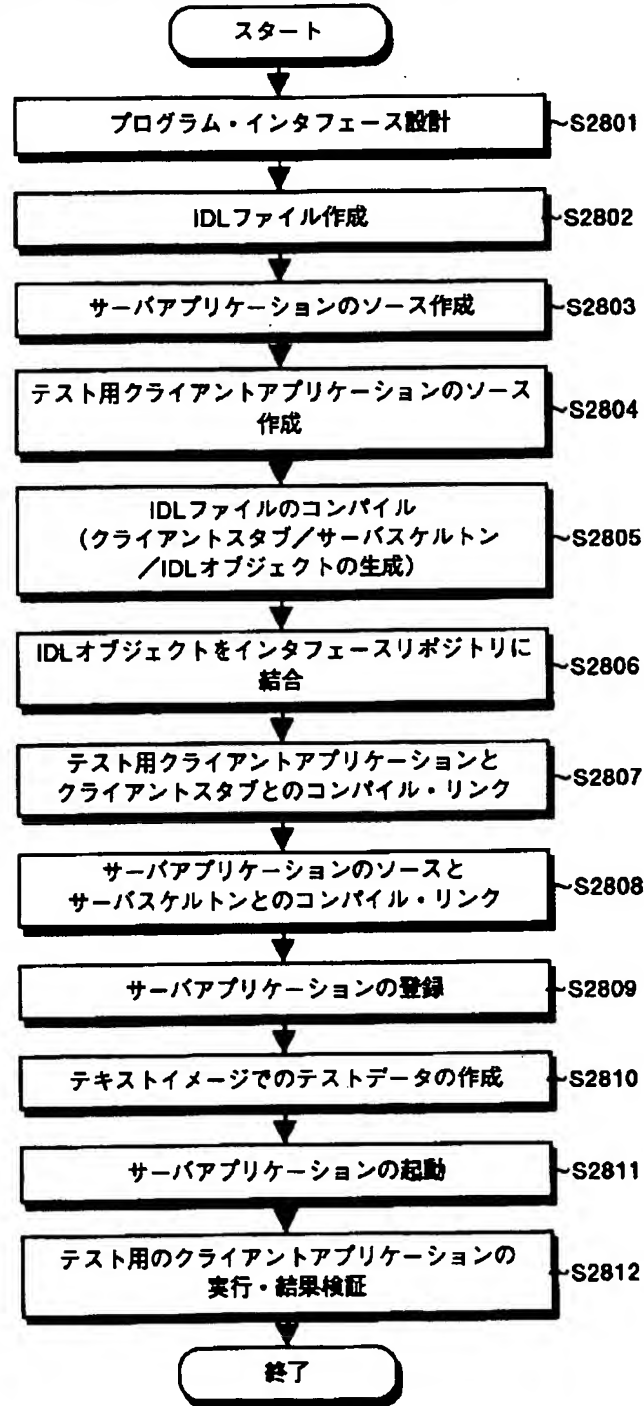
【図27】

従来のクライアントアプリケーション開発手順を示すフローチャート



【図28】

従来のサーバアプリケーション開発手順を示すフローチャート



フロントページの続き

Fターム(参考) 5B042 GA08 GA12 HH07 HH11
5B045 GG01 KK05
5B076 DD03 EC06